

# Intel® MPI Library for Linux\* OS

## Reference Manual

---

Copyright © 2003–2011 Intel Corporation

All Rights Reserved

Document Number: 315399-011

Revision: 4.0 Update 3

World Wide Web: <http://www.intel.com>

## Contents

1	About this Document .....	7
1.1	Intended Audience .....	7
1.2	Using Doc Type Field .....	7
1.3	Conventions and Symbols.....	8
1.4	Related Information.....	8
2	Command Reference .....	9
2.1	Compiler Commands.....	9
2.1.1	Compiler Command Options.....	9
2.1.2	Configuration Files.....	12
2.1.3	Profiles .....	13
2.1.4	Environment Variables .....	13
2.2	Simplified Job Startup Command .....	16
2.3	Scalable Process Management System (Hydra) Commands.....	18
2.3.1	Global Options .....	18
2.3.2	Local Options.....	26
2.3.3	Extended Device Control Options.....	27
2.3.4	Environment Variables .....	28
2.3.5	Cleaning up Utility.....	36
2.4	Multipurpose Daemon Commands.....	37
2.4.1	Job Startup Commands .....	44
2.4.2	Configuration Files.....	60
2.4.3	Environment Variables .....	60
2.5	Processor Information Utility.....	64
3	Tuning Reference .....	67
3.1	Automatic Tuning Utility .....	67
3.1.1	Cluster-specific Tuning.....	71
3.1.2	Application-specific Tuning.....	72
3.1.3	Tuning Utility Output .....	72
3.2	Process Pinning.....	72
3.2.1	Process Identification.....	72
3.2.2	Environment Variables .....	73
3.2.3	Interoperability with OpenMP* .....	79
3.3	Fabrics Control.....	88
3.3.1	Communication Fabrics Control .....	88
3.3.2	Shared Memory Control.....	94
3.3.3	DAPL-capable Network Fabrics Control .....	101
3.3.4	DAPL UD-capable Network Fabrics Control .....	109
3.3.5	TCP-capable Network Fabrics Control .....	117
3.3.6	TMI-capable Network Fabrics Control .....	119
3.3.7	OFA*-capable Network Fabrics Control .....	120
3.3.8	Failover Support in the OFA* Device .....	125
3.4	Dynamic Process Support .....	126
3.5	Collective Operation Control.....	127
3.5.1	I_MPI_ADJUST Family.....	127
3.5.2	I_MPI_MSG Family .....	130
3.6	Extended File System Support.....	134
3.6.1	Environment Variables .....	134
3.7	Compatibility Control .....	135
3.8	Miscellaneous .....	136
4	Statistics Gathering Mode .....	137
4.1	Native Statistics Format .....	137
4.2	IPM Statistics Format .....	142
4.2.1	Region Control .....	142
5	Fault Tolerance .....	150
5.1	Environment Variables .....	150
5.2	Usage Model.....	151

- 6 ILP64 Support ..... 152
  - 6.1 Using ILP64..... 152
  - 6.2 Known Issues and Limitations..... 152
- 7 Unified Memory Management ..... 153
- 8 Integration into Eclipse\* PTP..... 154
- 9 Glossary ..... 156
- 10 Index ..... 157

## Disclaimer and Legal Notices

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: <http://www.intel.com/design/literature.htm>

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. Go to:

[http://www.intel.com/products/processor\\_number/](http://www.intel.com/products/processor_number/)

MPEG-1, MPEG-2, MPEG-4, H.261, H.263, H.264, MP3, DV, VC-1, MJPEG, AC3, AAC, G.711, G.722, G.722.1, G.722.2, AMRWB, Extended AMRWB (AMRWB+), G.167, G.168, G.169, G.723.1, G.726, G.728, G.729, G.729.1, GSM AMR, GSM FR are international standards promoted by ISO, IEC, ITU, ETSI, 3GPP and other organizations. Implementations of these standards, or the standard enabled platforms may require licenses from various entities, including Intel Corporation.

BlueMoon, BunnyPeople, Celeron, Celeron Inside, Centrino, Centrino Inside, Cilk, Core Inside, E-GOLD, i960, Intel, the Intel logo, Intel AppUp, Intel Atom, Intel Atom Inside, Intel Core, Intel Inside, Intel Insider, the Intel Inside logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel Sponsors of Tomorrow., the Intel Sponsors of Tomorrow. logo, Intel StrataFlash, Intel vPro, Intel XScale, InTru, the InTru logo, the InTru Inside logo, InTru soundmark, Itanium, Itanium Inside, MCS, MMX, Moblin, Pentium, Pentium Inside, Puma, skool, the skool logo, SMARTi, Sound Mark, The Creators Project, The Journey Inside, Thunderbolt, Ultrabook, vPro Inside, VTune, Xeon, Xeon Inside, X-GOLD, XMM, X-PMU and XPOSYS are trademarks of Intel Corporation in the U.S. and other countries.

\* Other names and brands may be claimed as the property of others.

Microsoft, Windows, Visual Studio, Visual C++, and the Windows logo are trademarks, or registered trademarks of Microsoft Corporation in the United States and/or other countries.

Java is a registered trademark of Oracle and/or its affiliates.

Copyright (C) [2003]–[2011], Intel Corporation. All rights reserved.

## Optimization Notice

Intel compilers, associated libraries and associated development tools may include or utilize options that optimize for instruction sets that are available in both Intel and non-Intel microprocessors (for example SIMD instruction sets), but do not optimize equally for non-Intel microprocessors. In addition, certain compiler options for Intel compilers, including some that are not specific to Intel micro-architecture, are reserved for Intel microprocessors. For a detailed description of Intel compiler options, including the instruction sets and specific microprocessors they implicate, please refer to the "Intel Compiler User and Reference Guides" under "Compiler Options." Many library routines that are part of Intel compiler products are more highly optimized for Intel microprocessors than for other microprocessors. While the compilers and libraries in Intel compiler products offer optimizations for both Intel and Intel-compatible microprocessors, depending on the options you select, your code and other factors, you likely will get extra performance on Intel microprocessors.

Intel compilers, associated libraries and associated development tools may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include Intel® Streaming SIMD Extensions 2 (Intel® SSE2), Intel® Streaming SIMD Extensions 3 (Intel® SSE3), and Supplemental Streaming SIMD Extensions 3 (SSSE3) instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors.

While Intel believes our compilers and libraries are excellent choices to assist in obtaining the best performance on Intel and non-Intel microprocessors, Intel recommends that you evaluate other compilers and libraries to determine which best meet your requirements. We hope to win your business by striving to offer the best performance of any compiler or library; please let us know if you find we do not.

Notice revision #20110307

## Revision History

Document Number	Revision Number	Description	Revision Date
315399-001	3.1 Beta	Some new options and variables were added, three new sections "Statistics Gathering Mode", "Unified Memory Management", and "Integration into Eclipse* PTP" were created	/07/10/2007
315399-002	3.1	New names of variables were added, new section "Processor Information Utility" was added. Updated and reviewed for style	/10/02/2007
315399-003	3.1 build 038	Local options were added. Sections "Index", "Glossary", "Process Identification", and "Interoperability with OpenMP*" were added	/03/05/2008
315399-004	3.2	Sections "Process pinning", "Automatic Tuning Utility", and "Statistic Gathering Mode" were updated	/09/05/2008
315399-005	3.2 Update 1	Section "ILP64 Support" was added, section "Interoperability with OpenMP*" was updated	/03/04/2009
315399-006	4.0 Engineering Preview	Sections "Processor Information Utility", "Automatic Tuning Utility", and "Device Control" were updated	/06/24/2009
315399-007	4.0 Beta	Sections "Processor Information Utility", "Fabrics Control", "Statistics Gathering Mode", and "ILP64 Support" were updated. Section "Fault Tolerance" was added	/11/03/2009
315399-008	4.0	Sections "Fabrics Control", "Environment Variables", and "Job Startup Command" were updated.  Sections "Experimental Scalable Process Management System (Hydra)" and "Dynamic Process Support" were added.	/02/16/2010
315399-009	4.0 Update 1	Sections "Compiler Command Options", "Scalable Process Management System (Hydra)", "Job Startup Commands", "Multipurpose Daemon Commands", "Environment Variables", "Process Pinning", "Tuning Reference", "Shared Memory Control", "TCP-capable Network Fabrics Control" and "Disclaimer and Legal Notices" were updated.	/09/23/2010
315399-010	4.0 Update 2	Section "Disclaimer and Legal Notices" was updated.	/04/11/2011
315399-011	4.0 Update 3	Sections "Simplified Job Startup Command", "Scalable Process Management System (Hydra) Commands", "Job Startup Commands", "Processor Information Utility", "Automatic Tuning Utility", "Fabrics Control", "Collective Operation Control", and "Statistics Gathering Mode" were updated.  Sections "Cleaning up Utility", "Extended Device Control Options", and "IPM Statistics Format" were added.	/08/31/2011

# 1 About this Document

This *Reference Manual* provides you with a complete command and tuning reference for the Intel® MPI Library.

The Intel® MPI Library is a multi-fabric message passing library that implements the MessagePassing Interface, v2 (MPI-2) specification. It provides a standard library across Intel® platforms that enable adoption of MPI-2 functions as their needs dictate.

The Intel® MPI Library enables developers to change or to upgrade processors and interconnects as new technology becomes available without changes to the software or to the operating environment.

The library is provided in the following kits:

- *The Intel® MPI Library Runtime Environment* (RTO) has the tools you need to run programs, including multipurpose daemon\* (MPD) and supporting utilities, shared (.so) libraries, and documentation.
- *The Intel® MPI Library Development Kit* (SDK) includes all of the Runtime Environment components plus compilation tools, including compiler commands such as `mpiicc`, include files and modules, static (.a) libraries, debug libraries, trace libraries, and test codes.

## 1.1 Intended Audience

This *Reference Manual* helps an experienced user understand the full functionality of the Intel® MPI Library.

## 1.2 Using Doc Type Field

This *Reference Manual* contains the following sections

**Table 1.2-1 Document Organization**

Section	Description
Section 1 About this Document	Section 1 introduces this document
Section 2 Command Reference	Section 2 describes options and environment variables for compiler commands, job startup commands, and MPD daemon commands as well
Section 3 Tuning Reference	Section 3 describes environment variables used to influence program behavior and performance at run time
Section 4 Statistics Gathering Mode	Section 4 describes how to obtain statistics of MPI communication operations
Section 5 ILP64 Support	Section 5 describes support provided for the ILP64 programming model
Section 6 Unified Memory Management	Section 6 describes the unified memory management subsystem ( <code>i_malloc</code> )
Section 7 Integration into Eclipse* PTP	Section 7 describes the procedure for integration into Eclipse* Parallel Tools Platform
Section 8 Glossary	Section 8 explains basic terms used in this document

Section 9 Index	Section 9 references options and environment variables names
-----------------	--

## 1.3 Conventions and Symbols

The following conventions are used in this document.

**Table 1.3-1 Conventions and Symbols used in this Document**

<i>This type style</i>	Document or product names
<a href="#">This type style</a>	Hyperlinks
<b>This type style</b>	Commands, arguments, options, file names
<b>THIS_TYPE_STYLE</b>	Environment variables
<i>&lt;this type style&gt;</i>	Placeholders for actual values
[ items ]	Optional items
{ item   item }	Selectable items separated by vertical bar(s)
(SDK only)	For Software Development Kit (SDK) users only
<b>parameter</b>	Parameter name

## 1.4 Related Information

The following related documents that might be useful to the user:

[Product Web Site](#)

[Intel® MPI Library Support](#)

[Intel® Cluster Tools Products](#)

[Intel® Software Development Products](#)

## 2 Command Reference

### 2.1 Compiler Commands

(SDK only)

The following table lists available MPI compiler commands and the underlying compilers, compiler families, languages, and application binary interfaces (ABIs) that they support.

**Table 2.1-1 The Intel® MPI Library Compiler Drivers**

Compiler Command	Default Compiler	Supported Language(s)	Supported ABI(s)
<b>Generic Compilers</b>			
<code>mpicc</code>	<code>gcc, cc</code>	C	32/64 bit
<code>mpicxx</code>	<code>g++</code>	C/C++	32/64 bit
<code>mpifc</code>	<code>gfortran</code>	Fortran77*/Fortran 95*	32/64 bit
<b>GNU* Compilers Versions 3 and Higher</b>			
<code>mpigcc</code>	<code>gcc</code>	C	32/64 bit
<code>mpigxx</code>	<code>g++</code>	C/C++	32/64 bit
<code>mpif77</code>	<code>g77</code>	Fortran 77	32/64 bit
<code>mpif90</code>	<code>gfortran</code>	Fortran 95	32/64 bit
<b>Intel® Fortran, C++ Compilers Versions 10.0, 10.1, 11.0, 11.1 and Higher</b>			
<code>mpiicc</code>	<code>icc</code>	C	32/64 bit
<code>mpiicpc</code>	<code>icpc</code>	C++	32/64 bit
<code>mpiifort</code>	<code>ifort</code>	Fortran77/Fortran 95	32/64 bit

- Compiler commands are available only in the Intel® MPI Library Development Kit.
- Compiler commands are in the `<installdir>/<arch>/bin` directory. Where `<installdir>` refers to the Intel® MPI Library installation directory and `<arch>` is one of the following architectures:
  - `ia32` – IA-32 architecture binaries
  - `intel64` – Intel® 64 architecture binaries
- Ensure that the corresponding underlying compilers (32-bit or 64-bit, as appropriate) are already in your `PATH`.
- To port existing MPI-enabled applications to the Intel® MPI Library, recompile all sources.
- To display mini-help of a compiler command, execute it without any parameters.

#### 2.1.1 Compiler Command Options

##### `-mt_mpi`

Use this option to link the thread safe version of the Intel® MPI library at the following levels: `MPI_THREAD_FUNNELED`, `MPI_THREAD_SERIALIZED`, or `MPI_THREAD_MULTIPLE`.

The `MPI_THREAD_FUNNELED` level is provided by default by the thread safe version of the Intel® MPI library.

**NOTE:** If you specify either the `-openmp` or the `-parallel` options for the Intel® C Compiler, the thread safe version of the library is used.

**NOTE:** If you specify one of the following options for the Intel® Fortran Compiler, the thread safe version of the library is used:

- `-openmp`
- `-parallel`
- `-threads`
- `-reentrancy`
- `-reentrancy threaded`

## `-static_mpi`

Use this option to link the Intel® MPI library statically. This option does not affect the default linkage method for other libraries.

## `-static`

Use this option to link the Intel® MPI library statically. This option is passed to a compiler.

## `-config= <name>`

Use this option to source the configuration file. See [Configuration Files](#) for details.

## `-profile= <profile_name>`

Use this option to specify an MPI profiling library. The profiling library is selected using one of the following methods:

- Through the configuration file `<profile_name>.conf` located in the `<installdir>/<arch>/etc`. See [Profiles](#) for details.
- In the absence of the respective configuration file, by linking the library `lib<profile_name>.so` or `lib<profile_name>.a` located in the same directory as the Intel® MPI Library.

## `-t` or `-trace`

Use the `-t` or `-trace` option to link the resulting executable against the Intel® Trace Collector library. This has the same effect as if `-profile=vt` is used as an argument to `mpiicc` or another compiler script.

Use the `-t=log` or `-trace=log` option to link the resulting executable against the logging Intel® MPI Library and the Intel® Trace Collector libraries. The logging libraries trace internal Intel® MPI Library states in addition to the usual MPI function calls.

To use this option, include the installation path of the Intel® Trace Collector in the `VT_ROOT` environment variable. Set `I_MPI_TRACE_PROFILE` to the `<profile_name>` environment variable to specify another profiling library. For example, set `I_MPI_TRACE_PROFILE` to `vtfs` to link against the fail-safe version of the Intel® Trace Collector.

## **-check\_mpi**

Use this option to link the resulting executable against the Intel® Trace Collector correctness checking library. This has the same effect as if `-profile=vtmc` is used as an argument to `mpiicc` or another compiler script.

To use this option, include the installation path of the Intel® Trace Collector in the `VT_ROOT` environment variable. Set `I_MPI_CHECK_PROFILE` to the `<profile_name>` environment variable to specify another checking library.

## **-ilp64**

Use this option to enable ILP64 support. All integer arguments of the Intel MPI Library are treated as 64-bits values in this case.

**NOTE:** If you specify the `-i8` option for the Intel® Fortran Compiler, you still have to use the ILP64 option for linkage. See [ILP64 Support](#) for details.

## **-dynamic\_log**

Use this option in combination with the `-t` option to link in the Intel® Trace Collector library dynamically. This option does not affect the default linkage method for other libraries.

To run the resulting programs, include `$VT_ROOT/slib` in the `LD_LIBRARY_PATH` environment variable.

## **-g**

Use this option to compile a program in debug mode and link the resulting executable against the debugging version of the Intel® MPI Library. See [Environment variables](#), `I_MPI_DEBUG` for information on how to use additional debugging features with the `-g` builds.

## **-O**

Use this option to enable optimization.

## **-fast**

Use this Intel compiler option to maximize speed across the entire program. This option forces static linkage method for the Intel® MPI Library.

For implications on non-Intel processors, refer to the [xHost](#) documentation.

**NOTE:** It works for `mpiicc`, `mpicpc`, and `mpiifort` Intel compiler drivers only.

## **-echo**

Use this option to display everything that the command script does.

## **-show**

Use this option to learn how the underlying compiler is invoked. For example, use the following command to see the required compiler flags and options:

```
$ mpiicc -show -c test.c
```

Use the following command to see the required link flags, options, and libraries:

```
$ mpiicc -show -o a.out test.o
```

This is particularly useful for determining the command line for a complex build procedure that directly uses the underlying compilers.

**-{cc,cxx,fc,f77,f90}= <compiler>**

Use this option to select the underlying compiler.

For example, use the following command to select the Intel® C++ Compiler:

```
$ mpicc -cc=icc -c test.c
```

Make sure `icc` is in your path. Alternatively, you can specify the full path to the compiler.

**-gcc-version= <nnn>**

Use this option for compiler drivers `mpicxx` and `mpiicpc` when linking an application running in a particular GNU\* C++ environment. The valid `<nnn>` values are:

<nnn> value	GNU* C++ version
320	3.2.x
330	3.3.x
340	3.4.x
400	4.0.x
410	4.1.x
420	4.2.x
430	4.3.x

By default, the library compatible with the detected version of the GNU\* C++ compiler is used. Do not use this option if the GNU\* C++ version is older than 3.2.

**-compchk**

Use this option to enable compiler setup checks. In this case, each compiler driver performs checks to ensure that the appropriate underlying compiler is set up correctly.

**-v**

Use this option to print the compiler driver script version and its native compiler version.

## 2.1.2 Configuration Files

You can create compiler configuration files using the following file naming convention:

```
<installdir>/<arch>/etc/mpi<compiler>-<name>.conf
```

where:

- `<arch>` = {ia32,intel64} for the IA-32, and the Intel® 64 architectures
- `<compiler>` = {cc,cxx,f77,f90}, depending on the language being compiled
- `<name>` = name of underlying compiler with spaces replaced by hyphens

For example, the `<name>` value for `cc -64` is `cc--64`

Before compiling or linking to enable changes to the environment on a per compiler command basis, source these files or use the `-config` option, if it exists.

## 2.1.3 Profiles

You can select a profile library through the `-profile` option of the Intel® MPI Library compiler drivers. The profile files are located in the `<installdir>/<arch>/etc` directory. The Intel® MPI Library comes with several predefined profiles for the Intel® Trace Collector:

`<installdir>/etc/vt.conf` - regular Intel® Trace Collector library

`<installdir>/etc/vtfs.conf` - fail-safe Intel® Trace Collector library

`<installdir>/etc/vtmc.conf` – correctness checking Intel® Trace Collector library

You can also create your own profile as `<profile_name>.conf`

The following environment variables can be defined there:

`PROFILE_PRELIB` - libraries (and paths) to include before the Intel® MPI Library

`PROFILE_POSTLIB` - libraries to include after the Intel® MPI Library

`PROFILE_INCPATHS` - C preprocessor arguments for any include files

For instance, create a file `/myprof.conf` with the following lines:

`PROFILE_PRELIB="-L<path_to_myprof>/lib -lmyprof"`

`PROFILE_INCPATHS="-I<paths_to_myprof>/include"`

Use the command-line argument `-profile=myprof` for the relevant compile driver to select this new profile.

## 2.1.4 Environment Variables

### `I_MPI_{CC,CXX,FC,F77,F90}_PROFILE`

### `(MPI{CC,CXX,FC,F77,F90}_PROFILE)`

Specify a default profiling library.

#### Syntax

`I_MPI_{CC,CXX,FC,F77,F90}_PROFILE=<profile_name>`

#### Deprecated Syntax

`MPI{CC,CXX,FC,F77,F90}_PROFILE=<profile_name>`

#### Arguments

<code>&lt;profile_name&gt;</code>	Specify a default profiling library
-----------------------------------	-------------------------------------

#### Description

Set this environment variable to select a specific MPI profiling library to be used by default. This has the same effect as if `-profile=<profile_name>` were used as an argument to `mpiicc` or another Intel® MPI Library compiler driver.

### `I_MPI_TRACE_PROFILE`

Specify a default profile for the `-trace` option.

**Syntax**

`I_MPI_TRACE_PROFILE=<profile_name>`

**Arguments**

<code>&lt;profile_name&gt;</code>	Specify a tracing profile name. The default value is <code>vt</code>
-----------------------------------	--

**Description**

Set this environment variable to select a specific MPI profiling library to be used with the `-trace` option to `mpiicc` or another Intel® MPI Library compiler driver.

The `I_MPI_{CC,CXX,F77,F90}_PROFILE` environment variable overrides `I_MPI_TRACE_PROFILE`.

**I\_MPI\_CHECK\_PROFILE**

Specify a default profile for the `-check_mpi` option.

**Syntax**

`I_MPI_CHECK_PROFILE=<profile_name>`

**Arguments**

<code>&lt;profile_name&gt;</code>	Specify a checking profile name. The default value is <code>vtmc</code>
-----------------------------------	---

**Description**

Set this environment variable to select a specific MPI checking library to be used with the `-check_mpi` option to `mpiicc` or another Intel® MPI Library compiler driver.

The `I_MPI_{CC,CXX,F77,F90}_PROFILE` environment variable overrides `I_MPI_CHECK_PROFILE`.

**I\_MPI\_CHECK\_COMPILER**

Turn on/off compiler compatibility check.

**Syntax**

`I_MPI_CHECK_COMPILER=<arg>`

**Arguments**

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Enable checking the compiler
<code>disable   no   off   0</code>	Disable checking the compiler. This is the default value

**Description**

If `I_MPI_CHECK_COMPILER` is set to `enable`, the Intel MPI compiler drivers check the underlying compiler for compatibility. Normal compilation will be performed only if known version of underlying compiler is used.

**I\_MPI\_{CC,CXX,FC,F77,F90}****(MPICH\_{CC,CXX,FC,F77,F90})**

Set the path/name of the underlying compiler to be used.

**Syntax**

`I_MPI_{CC,CXX,FC,F77,F90}=<compiler>`

Deprecated Syntax

```
MPICH_{CC,CXX,FC,F77,F90}=<compiler>
```

Arguments

<compiler>	Specify the full path/name of compiler to be used
------------	---

Description

Set this environment variable to select a specific compiler to be used. Specify the full path to the compiler if it is not located in the search path.

**NOTE:** Some compilers may require additional command line options.

**NOTE:** The configuration file is sourced if it exists for a specified compiler. See [Configuration Files](#) for details.

I\_MPI\_ROOT

Set the Intel® MPI Library installation directory path.

Syntax

```
I_MPI_ROOT=<path>
```

Arguments

<path>	Specify the installation directory of the Intel® MPI Library
--------	--

Description

Set this environment variable to specify the installation directory of the Intel® MPI Library.

VT\_ROOT

Set Intel® Trace Collector installation directory path.

Syntax

```
VT_ROOT=<path>
```

Arguments

<path>	Specify the installation directory of the Intel® Trace Collector
--------	--

Description

Set this environment variable to specify the installation directory of the Intel® Trace Collector.

I\_MPI\_COMPILER\_CONFIG\_DIR

Set the location of the compiler configuration files.

Syntax

```
I_MPI_COMPILER_CONFIG_DIR=<path>
```

Arguments

<path>	Specify the location of the compiler configuration files. The default value is <installdir>/<arch>/etc
--------	--

Description

Set this environment variable to change the default location of the compiler configuration files.

## 2.2 Simplified Job Startup Command

### mpirun

#### Syntax

`mpirun <options>`

where `<options> := <mpiexec.hydra options> / [ <mpdboot options> ] <mpiexec options>`

#### Arguments

<code>&lt;mpiexec.hydra options&gt;</code>	<code>mpiexec.hydra</code> options as described below. This is the default operation mode.
<code>&lt;mpdboot options&gt;</code>	<code>mpdboot</code> options as described in the <code>mpdboot</code> command description below, except <code>-n</code>
<code>&lt;mpiexec options&gt;</code>	<code>mpiexec</code> options as described in the <code>mpiexec</code> section above

#### Description

Use this command to launch an MPI job. The `mpirun` command uses `Hydra` or `MPD` as underlying process managers. `Hydra` is the default process manager. Set the `I_MPI_PROCESS_MANAGER` environment variable to change the default value.

The `mpirun` command detects if the MPI job is submitted in a session allocated using a job scheduler like Torque\*, PBS Pro\*, OpenPBS\*, LSF\*, Parallelnavi\* NQS\*, SLURM\*, Sun\* Grid Engine\*, or LoadLeveler\*. In this case, the `mpirun` command extracts the host list from the respective environment and uses these nodes automatically according to the above scheme.

In this case, you do not need to create the `mpd.hosts` file. Allocate the session you need by using the particular job scheduler installed on your system, and use the `mpirun` command inside this session to run your MPI job.

When under batch system control, the `mpirun` command ignores the `-r | --rsh` option if `Hydra` is used as the underlying process manager. In this case, the corresponding bootstrap server is used. Use bootstrap specific options or corresponding environment variables explicitly to override the auto detected bootstrap server.

The `mpirun` command silently ignores the `MPD` specific options for compatibility reasons if you select `Hydra` as the active process manager. The following table provides the list of silently ignored and unsupported options. Avoid the usage of unsupported options if the `Hydra` process manager is used.

Ignored mpdboot Options	Ignored mpiexec Options	Unsupported mpdboot Options	Unsupported mpiexec Options
<code>--locons</code>	<code>-[g]envuser</code>	<code>--user=&lt;user&gt;   -u &lt;user&gt;</code>	<code>-a</code>
<code>--remcons</code>	<code>-[g]envexcl</code>	<code>--mpd=&lt;mpdcmd&gt;   -m &lt;mpdcmd&gt;</code>	
<code>--ordered   -o</code>	<code>-m</code>	<code>--shell   -s</code>	
<code>--maxbranch=&lt;maxbranch&gt;   -b &lt;maxbranch&gt;</code>	<code>-ifhn &lt;interface/hostname&gt;</code>	<code>-l</code>	
<code>--parallel-startup   -p</code>	<code>-ecfn &lt;filename&gt;</code>	<code>--ncpus=&lt;ncpus&gt;</code>	

Ignored mpdboot Options	Ignored mpiexec Options	Unsupported mpdboot Options	Unsupported mpiexec Options
	-tvsu		

If you select **MPD** as the process manager, the **mpirun** command starts an independent ring of **mpd** daemons, launch an MPI job, and shut down the **mpd** ring upon job termination.

The first non **mpdboot** option (including **-n** or **-np**) delimits the **mpdboot** and **mpiexec** options. All options up to this point, excluding the delimiting option, are passed to the **mpdboot** command. All options from this point on, including the delimiting option, are passed to the **mpiexec** command.

All configuration files and environment variables applicable to the **mpdboot** and **mpiexec** commands are also pertinent to **mpirun**.

The set of hosts is defined by the following rules, which are checked in this order:

1. All host names from the **mpdboot** host file (either **mpd.hosts** or the file specified by the **-f** option).
2. All host names returned by the **mpdtrace** command, if there is an **mpd** ring running.
3. Local host (a warning is issued in this case).

## I\_MPI\_MPIRUN\_CLEANUP

Control the environment cleanup.

### Syntax

**I\_MPI\_MPIRUN\_CLEANUP**=<value>

### Arguments

<value>	Define the option
enable   yes   on   1	Enable the environment cleanup
disable   no   off   0	Disable the environment cleanup. This is the default value

### Description

Use this environment variable to define whether to clean up the environment upon the **mpirun** completion.

## I\_MPI\_PROCESS\_MANAGER

Select a process manager to be used by the **mpirun** command.

### Syntax

**I\_MPI\_PROCESS\_MANAGER**=<value>

### Arguments

<value>	String value
hydra	Use Hydra process manager. This is the default value
mpd	Use MPD process manager

### Description

Set this environment variable to select the process manager to be used by the **mpirun** command.

**NOTE:** You can run each process manager directly by invoking the **mpiexec** command for MPD and the **mpiexec.hydra** command for Hydra.

# 2.3 Scalable Process Management System (Hydra) Commands

## mpiexec.hydra

The `mpiexec.hydra` is a more scalable alternative to the MPD process manager.

### Syntax

`mpiexec.hydra <g-options> <l-options> <executable>`

or

`mpiexec.hydra <g-options> <l-options> <executable1> : \`  
`<l-options> <executable2>`

### Arguments

<code>&lt;g-options&gt;</code>	Global options that apply to all MPI processes
<code>&lt;l-options&gt;</code>	Local options that apply to a single arg-set
<code>&lt;executable&gt;</code>	<code>./a.out</code> or <code>path/name</code> of the executable file

### Description

Use the `mpiexec.hydra` utility to run MPI applications without the MPD ring.

By using the first command-line syntax, you can start all MPI processes of the `<executable>` with the single arg-set. For example, the following command executes `a.out` over the specified `<# of processes>`:

```
$ mpiexec.hydra -f <hostsfile> -n <# of processes> ./a.out
```

`<hostsfile>` is the path/name of the file that has the list of machine names on which the application to run.

By using the second command-line syntax, you can start several MPI programs (or the same) with different arg-sets. For example, the following command executes two different binaries with different arg-sets:

```
$ mpiexec.hydra -f hosts.file -env <VAR1> <VAL1> -n 2 ./a.out : \  
-env <VAR2> <VAL2> -n 2 ./b.out
```

**NOTE:** If there is no "." in the `PATH` environment variable on all nodes in the cluster, specify `<executable>` as `./a.out` instead of `a.out`.

## 2.3.1 Global Options

### -hostfile <hostfile> or -f <hostsfile>

Use this option to specify machine names to run an application on. If a machine name is repeated, this name is used only once.

See also the [I\\_MPI\\_HYDRA\\_HOST\\_FILE](#) for more details.

**NOTE:** Use `-perhost`, `-rr` options to change the processes allocation on the cluster nodes.

**-machinefile <machine file> or -machine <machine file>**

Use this option to control the process placement through the *<machine file>*. The total number of processes to start is controlled by the *-n* option as usual.

See [-machinefile](#) option for more details.

**-genv <ENVVAR> <value>**

Use this option to set the *<ENVVAR>* environment variable to the specified *<value>* for all MPI processes.

**-genvall**

Use this option to enable propagation of all environment variables to all MPI processes.

**-genvnone**

Use this option to suppress propagation of any environment variables to any MPI processes.

**-genvlist <list of genv var names>**

Use this option to pass a list of environment variables with their current values. *<list of genv var names>* is a comma separated list of environment variables to be sent to the processes.

**-pmi-connect <mode>**

Use this option to choose the *PMI* message caching mode. Possible values are:

- *nocache* – do not cache *PMI* messages.
- *cache* – cache *PMI* messages on local *pmi\_proxy* management processes to minimize *PMI* requests. Cached information is propagated to the child management processes.
- *lazy-cache* – *cache* mode without propagation of the *PMI* information.

The *lazy-cache* mode is the default mode.

See [I\\_MPI\\_HYDRA\\_PMI\\_CONNECT](#) for more details.

**-perhost <# of processes >, -ppn <# of processes >, or -grr <# of processes>**

Use this option to place the indicated number of consecutive MPI processes on every host in group round robin fashion. See [I\\_MPI\\_PERHOST](#) for more details.

**-rr**

Use this option to place consecutive MPI processes on different hosts in round robin fashion. This option is equivalent to *-perhost 1*. See [I\\_MPI\\_PERHOST](#) for more details.

**(SDK only) -trace [<profiling\_library>] or -t [<profiling\_library>]**

Use this option to profile your MPI application using the indicated *<profiling\_library>*. If the *<profiling\_library>* is not mentioned, the default profiling library *libVT.so* is used.

Set the [I\\_MPI\\_JOB\\_TRACE\\_LIBS](#) environment variable to override the default profiling library.

**(SDK only) -check\_mpi [<checking\_library>]**

Use this option to check your MPI application using the indicated *<checking\_library>*. If *<checking\_library>* is not mentioned, the default checking library *libVTmc.so* is used.

Set the [I\\_MPI\\_JOB\\_CHECK\\_LIBS](#) environment variable to override the default checking library.

**-configfile <filename>**

Use this option to specify the file *<filename>* that contains command-line options. Blank lines and lines that start with '#' as the first character are ignored.

**-branch-count <num>**

Use this option to restrict the number of child management processes launched by the `mpiexec.hydra` command or each `pmi_proxy` management process.

See [I\\_MPI\\_HYDRA\\_BRANCH\\_COUNT](#) for more details.

**-pmi-aggregate or -pmi-noaggregate**

Use this option to switch on or off, respectively, the aggregation of the PMI requests. The default value is `-pmi-aggregate`, which means the aggregation is enabled by default.

See [I\\_MPI\\_HYDRA\\_PMI\\_AGGREGATE](#) for more details.

**-tv**

Use this option to run *<executable>* under the TotalView\* debugger. For example:

```
$ mpiexec.hydra -tv -n <# of processes> <executable>
```

See [Environment Variables](#) for information on how to select the TotalView\* executable file.

**NOTE:** Set the value of the `TVDSVRLAUNCHCMD` environment variable to `ssh` because the TotalView\* uses `rsh` by default.

**NOTE:** The TotalView\* debugger can display message queue state of your MPI program. To enable this feature, do the following steps:

1. Run your *<executable>* with the `-tv` option.

```
$ mpiexec.hydra-tv -n <# of processes> <executable>
```

2. Answer Yes to the question about stopping the `mpiexec.hydra` job.

To display the internal state of the MPI library textually, select the **Tools > Message Queue** command. If you select the **Process Window Tools > Message Queue Graph** command, the TotalView\* environment variable displays a window that shows a graph of the current message queue state. For more information, see the [TotalView\\*](#) environment variable.

**-tva <pid>**

Use this option to attach the TotalView\* debugger to an existing Intel® MPI job. Use the `mpiexec.hydra` process id as *<pid>*. For example:

```
$ mpiexec.hydra -tva <pid>
```

**-idb**

Use this option to run *<executable>* under the Intel® Debugger. For example:

```
$ mpiexec.hydra -idb -n <# of processes> <executable>
```

Include the installation path of the Intel® Debugger in the `IDB_HOME` environment variable.

By default, the Intel® Debugger runs in an Xterm terminal window. See the [I\\_MPI\\_HYDRA\\_IDB\\_TERMINAL](#) environment variable for information on how to select terminal for Intel® Debugger.

**-idba <pid>**

Use this option to attach the Intel® Debugger to an existing MPI job. Use the `mpiexec.hydra` process id as `<pid>`. For example:

```
$ mpiexec.hydra -idba <pid>
```

**-gdb**

Use this option to run `<executable>` under the GNU\* debugger. For example:

```
$ mpiexec.hydra-gdb -n <# of processes> <executable>
```

**-gdba <pid>**

Use this option to attach the GNU\* debugger to the existing Intel® MPI job. For example:

```
$ mpiexec.hydra-gdba <pid>
```

**-nolocal**

Use this option to avoid running `<executable>` on the host where the `mpiexec.hydra` is launched. You can use this option on clusters that deploy a dedicated master node for starting the MPI jobs and a set of dedicated compute nodes for running the actual MPI processes.

**-hosts <nodelist>**

Use this option to specify a particular `<nodelist>` on which the MPI processes are to be run. For example, the following commands run the executable `a.out` on hosts `host1` and `host2`:

```
$ mpiexec.hydra-n 2 -hosts host1,host2 ./a.out
```

**NOTE:** If `<nodelist>` consists of only one node, this option is interpreted as a local option. See [Local Options](#) for details.

**-iface <interface>**

Use this option to choose the appropriate network interface. For example, if the IP emulation of your InfiniBand\* network is configured on `ib0`, you can use `-iface ib0`.

See [I\\_MPI\\_HYDRA\\_IFACE](#) for more details.

**-demux <mode>**

Use this option to set `poll` or `select` polling mode for multiple I/O. The default is `poll`.

See [I\\_MPI\\_HYDRA\\_DEMUX](#) for more details.

**-enable-x or -disable-x**

Use this option to control the Xlib traffic forwarding. The default value is `-disable-x`, which means the Xlib traffic will not be forwarded.

**-l**

Use this option to insert the MPI process rank at the beginning of all lines written to the standard output.

**-tune [<arg >]**

where:

```
<arg> = {<dir_name>, <configuration_file>}.
```

Use this option to optimize the Intel® MPI Library performance through using the data collected by the `mpitune` utility.

If `<arg>` is not specified, the best-fit tune options are selected for the given configuration. The default location of the configuration file is `<installdir>/<arch>/etc` directory. You can override this default location by explicitly stating: `<arg>=<dir_name>`. The provided configuration file is used if you set `<arg>=<configuration_file>`.

### **-s <spec>**

Use this option to direct standard input to the specified MPI processes.

#### **Arguments**

<code>&lt;spec&gt;</code>	Define MPI process ranks
<code>all</code>	Use all processes
<code>&lt;l&gt;, &lt;m&gt;, &lt;n&gt;</code>	Specify an exact list and use processes <code>&lt;l&gt;</code> , <code>&lt;m&gt;</code> and <code>&lt;n&gt;</code> only. The default value is zero
<code>&lt;k&gt;, &lt;l&gt;-&lt;m&gt;, &lt;n&gt;</code>	Specify a range and use processes <code>&lt;k&gt;</code> , <code>&lt;l&gt;</code> through <code>&lt;m&gt;</code> , and <code>&lt;n&gt;</code>

### **-noconf**

Use this option to disable processing of the `mpiexec.hydra` configuration files described in [Configuration Files](#).

### **-ordered-output**

Use this option to avoid intermingling of data output from the MPI processes. This option affects both the standard output and the standard error streams.

**NOTE:** To use this option, end the last line output by each process with the end-of-line (`'\n'`) character. Otherwise the application may stop responding.

### **-path <directory>**

Use this option to specify the path to `<executable>`.

### **-cleanup**

Use this option to create a temporary file containing information about the launched processes. The file name is `mpiexec_${username}_$PPID.log`, where `PPID` is a parent process `PID`. This file is created in the temporary directory selected by the `-tmpdir` option. This file is used by the `mpicleanup` utility. If a job terminates successfully, the `mpiexec.hydra` command automatically removes this file.

See [I MPI HYDRA CLEANUP](#) for more details.

### **-tmpdir**

Use this option to set a directory for temporary files.

See [I MPI TMPDIR](#) for more details.

### **-version or -V**

Use this option to display the version of the Intel® MPI Library.

### 2.3.1.1 Bootstrap Options

#### -bootstrap <bootstrap server>

Use this option to select a built-in bootstrap server to use. A bootstrap server is the basic remote node access mechanism that is provided on any system. Hydra supports multiple runtime bootstrap servers such as `ssh`, `rsh`, `fork`, `slurm`, `ll`, `lsf`, `sge`, and `jmi` to launch processes. The default bootstrap server is `ssh`. By setting `slurm`, `ll`, `lsf`, or `sge`, you use the corresponding `srun`, `llspawn.stdio`, `blaunch`, and `qrsh` internal job scheduler utilities to launch service processes under a particular job scheduler.

To enable the tight integration with SLURM\*, use the `jmi` bootstrap server. Tight integration provides registering processes identifiers to job schedulers. These identifiers count used resources by a particular job and cleanup the nodes if a job termination occurs.

See the [-bootstrap jmi](#) description and the [I\\_MPI\\_HYDRA\\_BOOTSTRAP](#) environment variable for details.

#### -bootstrap-exec <bootstrap server>

Use this option to set the executable to be used as a bootstrap server. Possible values are `ssh`, `rsh`, `fork`, and `slurm`. The default bootstrap server is `ssh`. For example:

```
$ mpiexec.hydra -bootstrap ssh -bootstrap-exec /usr/bin/ssh \
-f hosts.file -env <VAR1> <VAL1> -n 2 ./a.out
```

See [I\\_MPI\\_HYDRA\\_BOOTSTRAP](#) for more details.

#### -bootstrap jmi

Use this option to enable tight integration with SLURM\* job schedulers. The tight integration is implemented by using particular job scheduler API or utility. If you specify this option, the default `libjmi.so` library is loaded. You can overwrite the default library name through the `I_MPI_HYDRA_JMI_LIBRARY` environment variable.

See [I\\_MPI\\_HYDRA\\_JMI\\_LIBRARY](#) for more details.

### 2.3.1.2 Binding Options

#### -binding

Use this option to pin MPI processes to a particular CPU and avoid undesired process migration. In the following syntax, the quotes may be omitted for one-member list. Each parameter is responsible for a single pinning property.

This option is available for both Intel® and non-Intel microprocessors, but it may perform additional optimizations for Intel microprocessors than it performs for non-Intel microprocessors.

#### Syntax

```
-binding "<parameter>=<value>[;<parameter>=<value> ...]"
```

#### Parameters

<code>pin</code>	Pinning switch
<code>enable</code>   <code>yes</code>   <code>on</code>   <code>1</code>	Turn on the pinning property. This is the default value

<code>disable   no   off   0</code>	Turn off the pinning property
-------------------------------------	-------------------------------

<code>cell</code>	Pinning resolution power
<code>unit</code>	Basic processor unit (logical CPU)
<code>core</code>	Processor core in multi-core system

<code>map</code>	Determined mapping
<code>spread</code>	The processes are mapped consecutively to separate processor cells. Thus, the processes do not share the common resources by the adjacent cells.
<code>scatter</code>	The processes are mapped to separate processor cells. Adjacent processes are mapped on the cells that are the most remote according to multi-core topology.
<code>bunch</code>	The processes are mapped to separate processor cells by #processes/#sockets processes per socket. Each socket processor portion is a set of the cells that are the closest according to multi-core topology.
<code>p0,p1,...,pn</code>	The processes are mapped on the separate processors in the way that the processors are specified in the <code>p0,p1,...,pn</code> list: <i>i</i> <sup>th</sup> process is mapped on <i>p<sub>i</sub></i> processor <i>p<sub>i</sub></i> is one of the following values: alone processor number like <i>n</i> , range of processor numbers like <i>n-m</i> , <i>-1</i> The <i>-1</i> value means no pinning for corresponding process.
<code>[m0,m1,...,mn]</code>	<i>i</i> <sup>th</sup> process is mapped on the processor subset that is defined by <i>m<sub>i</sub></i> hexadecimal mask using the following rule: <i>j</i> <sup>th</sup> processor is included into <i>m<sub>i</sub></i> subset if <i>j</i> <sup>th</sup> bit of <i>m<sub>i</sub></i> equals to 1.

<code>domain</code>	Processor domain set on a node
<code>cell</code>	Each domain of the set is a single processor cell (unit or core)
<code>core</code>	Each domain of the set consists of the processor cells that share a particular core.
<code>cache1</code>	Each domain of the set consists of the processor cells that share a particular level 1 cache.
<code>cache2</code>	Each domain of the set consists of the processor cells that share a particular level 2 cache.
<code>cache3</code>	Each domain of the set consists of the processor cells that share a particular level 3 cache.

<b>cache</b>	The set whose elements are the largest domains among <b>cache1</b> , <b>cache2</b> , and <b>cache3</b>
<b>socket</b>	Each domain of the set consists of the processor cells that are located on a particular socket.
<b>node</b>	All processor cells on a node are arranged into a single domain.
<b>&lt;size&gt;[:&lt;layout&gt;]</b>	<p>Each domain of the set consists of <b>&lt;size&gt;</b> processor cells. <b>&lt;size&gt;</b> may have the following values:</p> <p><b>auto</b> – domain size = #cells/#processes</p> <p><b>omp</b> – domain size = <b>OMP_NUM_THREADS</b> environment variable value</p> <p>positive integer – exact value of domain size</p> <p>Domain size is limited by the node size.</p> <p>Each member location inside the domain is defined by the optional <b>&lt;layout&gt;</b> parameter value:</p> <p><b>compact</b> – as close with others as possible according to multi-core topology</p> <p><b>scatter</b> – as far away from others as possible according to multi-core topology</p> <p><b>range</b> – by BIOS numbering of processors</p> <p>If <b>&lt;layout&gt;</b> parameter is omitted, <b>compact</b> is assumed as the value of <b>&lt;layout&gt;</b>.</p>

<b>order</b>	Linear ordering of the domains
<b>compact</b>	Order the domain set so that adjacent domains are the closest according to multi-core topology
<b>scatter</b>	Order the domain set so that adjacent domains are the most remote according to multi-core topology
<b>range</b>	Order the domain set according to processor BIOS numbering

<b>offset</b>	Domain list offset
<b>&lt;n&gt;</b>	Integer number that indicates the starting domain among the linear ordered domains. This domain gets zero number. The numbers of other domains will be cyclic shifted conformably

### 2.3.1.3 Communication Subsystem Options

#### -rmk <RMK>

Use this option to select resource management kernel to be used. Intel® MPI Library only supports **pbs**.

See [1 MPI\\_HYDRA\\_RMK](#) for more details.

### 2.3.1.4 Other Options

#### -verbose

Use this option to print debug information from **mpiexec.hydra**, such as:

- Service processes arguments
- Environment variables and arguments passed to start an application
- PMI requests/responses during a job life cycle

See [I\\_MPI\\_HYDRA\\_DEBUG](#) for more details.

### **-print-rank-map**

Use this option to print rank mapping.

### **-print-all-exitcodes**

Use this option to print exit codes of all processes.

## 2.3.2 Local Options

### **-n <# of processes> or -np <# of processes>**

Use this option to set the number of MPI processes to run with the current arg-set.

### **-env <ENVVAR> <value>**

Use this option to set the *<ENVVAR>* environment variable to the specified *<value>* for all MPI processes in the current arg-set.

### **-envall**

Use this option to propagate all environment variables in the current environment.

See [I\\_MPI\\_HYDRA\\_ENV](#) for more details.

### **-envnone**

Use this option to suppress propagation of any environment variables to the MPI processes in the current arg-set.

### **-envlist <list of env var names>**

Use this option to pass a list of environment variables with their current values. *<list of env var names>* is a comma separated list of environment variables to be sent to the processes.

### **-host <nodename>**

Use this option to specify a particular *<nodename>* on which the MPI processes are to be run. For example, the following command executes *a.out* on hosts *host1* and *host2*:

```
$ mpiexec.hydra -n 2 -hosts host1 ./a.out : -n 2 -hosts host2 ./a.out
```

### **-path <directory>**

Use this option to specify the path to *<executable>* to be run in the current arg-set.

### **-wdir <directory>**

Use this option to specify the working directory in which *<executable>* runs in the current arg-set.

**-umask <umask>**

Use this option to perform the `umask <umask>` command for the remote process.

## 2.3.3 Extended Device Control Options

**-rdma**

Use this option to select an RDMA-capable network fabric for inter-nodes communication. The application attempts to use first available RDMA-capable network fabric from the list `dapl` or `ofa`. If no such fabric is available, other fabrics from the list `tcp` or `tmi` are used. This option is equivalent to the `-genv I_MPI_FABRICS_LIST dapl,ofa,tcp,tmi -genv I_MPI_FALLBACK 1` setting.

**-RDMA**

Use this option to select an RDMA-capable network fabric for inter-nodes communication. The application attempts to use first available RDMA-capable network fabric from the list `dapl` or `ofa`. The application fails if no such fabric is found. This option is equivalent to the `-genv I_MPI_FABRICS_LIST dapl,ofa -genv I_MPI_FALLBACK 1` setting.

**-dapl**

Use this option to select DAPL capable network fabric for inter-nodes communication. The application attempts to use DAPL capable network fabric. If no such fabric is available, another fabric from the list `tcp`, `tmi` or `ofa` is used. This option is equivalent to the `-genv I_MPI_FABRICS_LIST dapl,tcp,tmi,ofa -genv I_MPI_FALLBACK 1` setting.

**-DAPL**

Use this option to select DAPL capable network fabric for inter-nodes communication. The application fails if no such fabric is found. This option is equivalent to the `-genv I_MPI_FABRICS_LIST dapl -genv I_MPI_FALLBACK 0` setting.

**-ib**

Use this option to select OFA capable network fabric for inter-nodes communication. The application attempts to use OFA capable network fabric. If no such fabric is available, another fabrics from the list `dapl`, `tcp` or `tmi` is used. This option is equivalent to the `-genv I_MPI_FABRICS_LIST ofa,dapl,tcp,tmi -genv I_MPI_FALLBACK 1` setting.

**-IB**

Use this option to select OFA capable network fabric for inter-nodes communication. The application fails if no such fabric is found. This option is equivalent to the `-genv I_MPI_FABRICS_LIST ofa -genv I_MPI_FALLBACK 0` setting.

**-tmi**

Use this option to select TMI capable network fabric for inter-nodes communication. The application attempts to use TMI capable network fabric. If no such fabric is available, another fabric from the list `dapl`, `tcp` or `ofa` is used. This option is equivalent to the `-genv I_MPI_FABRICS_LIST tmi,dapl,tcp,ofa -genv I_MPI_FALLBACK 1` setting.

**-TMI**

Use this option to select TMI capable network fabric for inter-nodes communication. The application fails if no such fabric is found. This option is equivalent to the `-genv I_MPI_FABRICS_LIST tmi -genv I_MPI_FALLBACK 0` setting.

## -mx

Use this option to select Myrinet MX\* network fabric for inter-nodes communication. The application attempts to use Myrinet MX\* network fabric. If no such fabric is available, another fabrics from the list `dapl,tcp` or `ofa` is used. This option is equivalent to the `-genv I_MPI_FABRICS_LIST tmi,dapl,tcp,ofa -genv I_MPI_TMI_PROVIDER mx -genv I_MPI_DAPL_PROVIDER mx -genv I_MPI_FALLBACK 1` setting.

## -MX

Use this option to select Myrinet MX\* network fabric for inter-nodes communication. The application fails if no such fabric is found. This option is equivalent to the `-genv I_MPI_FABRICS_LIST tmi -genv I_MPI_TMI_PROVIDER mx -genv I_MPI_FALLBACK 0` setting.

## -psm

Use this option to select Qlogic\* network fabric for inter-nodes communication. The application attempts to use Qlogic\* network fabric. If no such fabric is available, another fabrics from the list `dapl,tcp` or `ofa` is used. This option is equivalent to the `-genv I_MPI_FABRICS_LIST tmi,dapl,tcp,ofa -genv I_MPI_TMI_PROVIDER psm -genv I_MPI_FALLBACK 1` setting.

## -PSM

Use this option to select Qlogic\* network fabric for inter-nodes communication. The application fails if no such fabric is found. This option is equivalent to the `-genv I_MPI_FABRICS_LIST tmi -genv I_MPI_TMI_PROVIDER psm -genv I_MPI_FALLBACK 0` setting.

## -gm

Use this option to select Myrinet\* GM\* network fabric for inter-nodes communication. This option is equivalent to the `-genv I_MPI_DEVICE rdssm:GmHca0 -genv I_MPI_FALLBACK_DEVICE 1` setting.

**NOTE:** This environment variable is deprecated and supported mostly for backward compatibility.

## -GM

Use this option to select Myrinet\* GM\* network fabric for inter-nodes communication. The application fails if no such fabric is found. This option is equivalent to the `-genv I_MPI_DEVICE rdssm:GmHca0 -genv I_MPI_FALLBACK_DEVICE 0` setting.

**NOTE:** This environment variable is deprecated and supported mostly for backward compatibility.

## 2.3.4 Environment Variables

### I\_MPI\_HYDRA\_HOST\_FILE

Set the hosts file to run the application.

#### Syntax

`I_MPI_HYDRA_HOST_FILE=<arg>`

#### Deprecated Syntax

`HYDRA_HOST_FILE=<arg>`

#### Arguments

<code>&lt;arg&gt;</code>	String parameter
<code>&lt;hostsfile&gt;</code>	Full or relative path to hosts file

**Description**

Set this environment variable to specify the hosts file.

**I\_MPI\_HYDRA\_DEBUG**

Print out the debug information.

**Syntax**

`I_MPI_HYDRA_DEBUG=<arg>`

**Arguments**

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Turn on the debug output
<code>disable   no   off   0</code>	Turn off the debug output. This is the default value

**Description**

Set this environment variable to `1` to enable the debug mode and `0` to turn off the debug mode.

**I\_MPI\_HYDRA\_ENV**

Set it to `all` to pass the environment.

**Syntax**

`I_MPI_HYDRA_ENV=<arg>`

**Arguments**

<code>&lt;arg&gt;</code>	String parameter
<code>all</code>	Full or relative path to hosts file

**Description**

By default, the launching node environment is passed to the executables as long as it does not overwrite any of the environment variables that have been preset by the remote shell.

**I\_MPI\_JOB\_TIMEOUT, I\_MPI\_MPIEXEC\_TIMEOUT****(MPIEXEC\_TIMEOUT)**

Set the timeout period for `mpiexec.hydra`.

**Syntax**

`I_MPI_JOB_TIMEOUT=<timeout>`

`I_MPI_MPIEXEC_TIMEOUT=<timeout>`

**Deprecated Syntax**

`MPIEXEC_TIMEOUT=<timeout>`

**Arguments**

<code>&lt;timeout&gt;</code>	Define <code>mpiexec.hydra</code> timeout period in seconds
<code>&lt;n&gt; &gt;= 0</code>	The default timeout value is zero, which means no timeout

Description

Set this environment variable to make `mpiexec.hydra` terminate the job in `<timeout>` seconds after its launch. The `<timeout>` value should be greater than zero. Otherwise the environment variable setting is ignored.

**NOTE:** Set the `I_MPI_JOB_TIMEOUT` environment variable in the shell environment before executing the `mpiexec.hydra` command. Do not use the `-genv` or `-env` options to set the `<timeout>` value. Those options are used only for passing environment variables to the MPI process environment.

I\_MPI\_JOB\_TIMEOUT\_SIGNAL

(MPIEXEC\_TIMEOUT\_SIGNAL)

Define the signal when a job is terminated because of a timeout.

Syntax

`I_MPI_JOB_TIMEOUT_SIGNAL= <number>`

Deprecated Syntax

`MPIEXEC_TIMEOUT_SIGNAL= <number>`

Arguments

<code>&lt;number&gt;</code>	Define signal number
<code>&lt;n&gt; &gt; 0</code>	The default value is 9 (SIGKILL)

Description

Define a signal number for the stop of a task if the timeout period specified by `I_MPI_JOB_TIMEOUT` expires. If you set a signal number unsupported by the system, `mpiexec.hydra` prints a warning message and continues task termination using the default signal number 9 (SIGKILL).

I\_MPI\_JOB\_ABORT\_SIGNAL

Define a signal to be sent to all processes when a job is terminated unexpectedly.

Syntax

`I_MPI_JOB_ABORT_SIGNAL= <number>`

Arguments

<code>&lt;number&gt;</code>	Define signal number
<code>&lt;n&gt; &gt; 0</code>	The default value is 9 (SIGKILL)

Description

Set this environment variable to define a signal for task termination. If you set an unsupported signal number, `mpiexec.hydra` prints a warning message and uses the default signal 9 (SIGKILL).

I\_MPI\_JOB\_SIGNAL\_PROPAGATION

(MPIEXEC\_SIGNAL\_PROPAGATION)

Control signal propagation.

**Syntax**

```
I_MPI_JOB_SIGNAL_PROPAGATION=<arg>
```

**Deprecated Syntax**

```
MPIEXEC_SIGNAL_PROPAGATION=<arg>
```

**Arguments**

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Turn on propagation
<code>disable   no   off   0</code>	Turn off propagation. This is the default value

**Description**

Set this environment variable to control propagation of the signals (`SIGINT`, `SIGALRM`, and `SIGTERM`) that may be received by the `MPD` daemons. If you enable signal propagation, the received signal is sent to all processes of the MPI job. If you disable signal propagation, all processes of the MPI job are stopped with the default signal 9 (`SIGKILL`).

**I\_MPI\_HYDRA\_BOOTSTRAP**

Set the bootstrap server.

**Syntax**

```
I_MPI_HYDRA_BOOTSTRAP=<arg>
```

**Arguments**

<code>&lt;arg&gt;</code>	String parameter
<code>ssh   rsh   fork   slurm   ll   lsf   sge   jmi</code>	The remote node access mechanism. The default is <code>ssh</code>

**Description**

Set this environment variable to specify the bootstrap server.

**I\_MPI\_HYDRA\_BOOTSTRAP\_EXEC**

Set the executable to be used as bootstrap server.

**Syntax**

```
I_MPI_HYDRA_BOOTSTRAP_EXEC=<arg>
```

**Arguments**

<code>&lt;arg&gt;</code>	String parameter
<code>ssh   rsh   fork   slurm   ll   lsf   sge   jmi</code>	The remote node access mechanism. The default is <code>ssh</code>

**Description**

Set this environment variable to specify the executable to be used as bootstrap server.

**I\_MPI\_HYDRA\_RMK**

Use the resource management kernel.

**Syntax**

`I_MPI_HYDRA_RMK=<arg>`

**Arguments**

<code>&lt;arg&gt;</code>	String parameter
<code>&lt;rmk&gt;</code>	Resource management kernel. The only supported value is <code>pbs</code>

**Description**

Set this environment variable to use resource management kernel. Intel® MPI Library only supports `pbs`.

**I\_MPI\_HYDRA\_PMI\_CONNECT**

Define `PMI` messages processing method.

**Syntax**

`I_MPI_HYDRA_PMI_CONNECT=<value>`

**Arguments**

<code>&lt;value&gt;</code>	An algorithm to be used
<code>nocache</code>	Do not cache <code>PMI</code> messages
<code>cache</code>	Cache <code>PMI</code> messages on local <code>pmi_proxy</code> management processes to minimize <code>PMI</code> requests. Cached information is propagated to child management processes
<code>lazy-cache</code>	<code>cache</code> mode without propagation. This is default value

**Description**

Use this environment variable to select the `PMI` messages processing method. If `-pmi-connect` is explicitly presented in the `mpiexec.hydra` command line, `I_MPI_HYDRA_PMI_CONNECT` is ignored.

**I\_MPI\_PERHOST**

Define the default settings for the `-perhost` option in the `mpiexec` and `mpiexec.hydra` command.

**Syntax**

`I_MPI_PERHOST=<value>`

**Arguments**

<code>&lt;value&gt;</code>	Define a value that is used for the <code>-perhost</code> option by default
<code>integer &gt; 0</code>	Exact value for the option
<code>all</code>	All logical CPUs on a node
<code>allcores</code>	All cores (physical CPUs) on a node

**Description**

Set this environment variable to define the default setting for the `-perhost` option. If `-perhost` is explicitly presented in the command line, `I_MPI_PERHOST` has no effect. The `-perhost` option is assumed with its value if `I_MPI_PERHOST` is defined.

## I\_MPI\_JOB\_TRACE\_LIBS

Choose the libraries to preload through the `-trace` option.

### Syntax

`I_MPI_JOB_TRACE_LIBS=<arg>`

### Deprecated Syntax

`MPIEXEC_TRACE_LIBS=<arg>`

### Arguments

<code>&lt;arg&gt;</code>	String parameter
<code>&lt;list&gt;</code>	Blank separated list of libraries to preload. The default value is <code>vt</code>

### Description

Set this environment variable to choose an alternative library for preloading through the `-trace` option.

## I\_MPI\_JOB\_CHECK\_LIBS

Choose the libraries to preload through the `-check_mpi` option.

### Syntax

`I_MPI_JOB_CHECK_LIBS=<arg>`

### Arguments

<code>&lt;arg&gt;</code>	String parameter
<code>&lt;list&gt;</code>	Blank separated list of libraries to preload. The default value is <code>vtmc</code>

### Description

Set this environment variable to choose an alternative library for preloading through the `-check_mpi` option.

## I\_MPI\_HYDRA\_BRANCH\_COUNT

Set branch count.

### Syntax

`I_MPI_HYDRA_BRANCH_COUNT = <num>`

### Arguments

<code>&lt;num&gt;</code>	Number
<code>&lt;n&gt; &gt;= 0</code>	Default value <ul style="list-style-type: none"> <li><code>-1</code> if less than 128 nodes used. It means do not use hierarchical structure</li> <li><code>32</code>, if more than 127 nodes used</li> </ul>

### Description

Set this environment variable to restrict the number of child management processes launched by `mpiexec.hydra` or each `pmi_proxy` management process.

## I\_MPI\_HYDRA\_PMI\_AGGREGATE

Turn on/off **PMI** messages aggregation.

### Syntax

**I\_MPI\_HYDRA\_PMI\_AGGREGATE**=<arg>

### Arguments

<arg>	Binary indicator
enable   yes   on   1	Enable <b>PMI</b> messages aggregation. This is the default value
disable   no   off   0	Disable <b>PMI</b> messages aggregation

### Description

Set this environment variable to enable/disable **PMI** messages aggregation.

## I\_MPI\_HYDRA\_IDB\_TERMINAL

Set the terminal emulator for Intel® Debugger.

### Syntax

**I\_MPI\_HYDRA\_IDB\_TERMINAL**=<arg>

### Arguments

<arg>	String parameter
xterm	Select Xterm terminal emulator. This is default value
screen	Select screen terminal emulator

### Description

Set this environment variable to specify the terminal emulator for Intel® Debugger.

## I\_MPI\_HYDRA\_GDB\_REMOTE\_SHELL

Set the remote shell command to run GNU\* debugger.

### Syntax

**I\_MPI\_HYDRA\_GDB\_REMOTE\_SHELL**= <arg>

### Arguments

<arg>	String parameter
ssh	Select Secure Shell (SSH). This is default value
rsh	Select Remote shell (RSH)

### Description

Set this environment variable to specify the remote shell command to run GNU\* debugger on non-local machines. You can use this environment variable to specify any command that has the same syntax as SSH or RSH.

## I\_MPI\_HYDRA\_JMI\_LIBRARY

Define the default setting of **JMI** library to be used by Hydra process manager.

## Syntax

`I_MPI_JMI_ HYDRA_LIBRARY=<value>`

## Arguments

<code>&lt;value&gt;</code>	Define a string value, name, or path to <b>JMI</b> dynamic library
<code>Libjmi_slurm.so.1.0</code>	Set the library name or full path to library name. The default value is <code>libjmi.so</code>

## Description

Set this environment variable to define the **JMI** library to be loaded by Hydra processor manager. Set the full path to the library if it is not to `LD_LIBRARY_PATH` environment variable. If `mpirun` is used, the **JMI** library is automatically detected and set. You do not need to set this environment variable.

## I\_MPI\_HYDRA\_IFACE

Set the network interface.

## Syntax

`I_MPI_HYDRA_IFACE= <arg>`

## Arguments

<code>&lt;arg&gt;</code>	String parameter
<code>&lt;network interface&gt;</code>	The network interface configured in your system

## Description

Set this environment variable to specify the network interface to use. For example, use `-iface ib0`, if the IP emulation of your **InfiniBand** network is configured on `ib0`.

## I\_MPI\_HYDRA\_DEMUX

Set the **demux** mode.

## Syntax

`I_MPI_HYDRA_DEMUX= <arg>`

## Arguments

<code>&lt;arg&gt;</code>	String parameter
<code>poll   select</code>	Multiple I/O <b>demux</b> mode engine

## Description

Set this environment variable to specify the multiple I/O **demux** mode engine. The default is `Poll`.

## I\_MPI\_HYDRA\_CLEANUP

Control the creation of the default **mpicleanup** input file.

## Syntax

`I_MPI_HYDRA_CLEANUP= <value>`

## Arguments

<code>&lt;value&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Enable the <b>mpicleanup</b> input file creation

<code>disable   no   off   0</code>	Disable the <code>mpicleanup</code> input file creation. This is the default value
-------------------------------------	--

### Description

Set the `I_MPI_HYDRA_CLEANUP` environment variable to create the input file for the `mpicleanup` utility.

## I\_MPI\_TMPDIR

### (TMPDIR)

Set the temporary directory.

### Syntax

`I_MPI_TMPDIR=<arg>`

### Arguments

<code>&lt;arg&gt;</code>	String parameter
<code>&lt;path&gt;</code>	Set the temporary directory. The default value is <code>/tmp</code>

### Description

Set this environment variable to specify the temporary directory to store the `mpicleanup` input file.

## 2.3.5 Cleaning up Utility

### mpicleanup

Clean up the environment after an abnormally terminated MPI run under the `mpiexec.hydra` process manager.

### Syntax

```
mpicleanup [ -i <input_file> | -t -f <hostsfile> ] [ -r <rshcmd> ] \
           [ -b <branch_count> ] [-p] [-s | -d] [-h] [-V]
```

or

```
mpicleanup [ --input <input_file> | --total --file <hostsfile> ] \
           [ --rsh <rshcmd> ] [ --branch <branch_count> ] [ --parallel ] \
           [ --silent | --verbose ] [ --help ] [ --version ]
```

### Arguments

<code>-i &lt;input_file&gt;   --input &lt;input_file&gt;</code>	Specify the input file generated by <code>mpiexec.hydra</code> . The default value is <code>mpiexec_\${username}_\$PPID.log</code> located in the temporary directory regulated by <code>I_MPI_TMPDIR</code> , <code>TMPDIR</code> or <code>/tmp</code> directory
<code>-t   --total</code>	Use the total mode to stop all user processes on specified machines. This option is not applicable for the <code>root</code> user
<code>-f &lt;hostsfile&gt;   --file &lt;hostsfile&gt;</code>	Specify the file containing a list of machines to clean up on
<code>-r &lt;rshcmd&gt; /</code>	Specify the remote shell to use. The default shell is <code>ssh</code>

<code>--rsh &lt;rshcmd&gt;</code>	
<code>-b &lt;branch_count&gt;   --branch &lt;branch_count&gt;</code>	Specify a machines number to restrict the number of the child processes. The default value is <code>32</code>
<code>-p   --parallel</code>	Use the parallel launch mode. This option is only applicable if all hosts are available. Otherwise a part of machines may stay in an undetermined state
<code>-s   --silent</code>	Suppress extra output generation
<code>-d   --verbose</code>	Output verbose information
<code>-h   --help</code>	Display a help message
<code>-V   --version</code>	Display Intel® MPI Library version information

### Description

Use this command to clean up the environment after an abnormal MPI job termination.

For example, use the following command to stop processes mentioned in the input file generated by the prior `mpiexec.hydra` invocation:

```
$ mpicleanup
```

or

```
$ mpicleanup --input /path/to/input.file
```

Use the following command to stop all user's processes on the machines specified in the `hostsfile` file:

```
$ mpicleanup --file hostsfile --total
```

## 2.4 Multipurpose Daemon Commands

### mpd

Start `mpd` daemon.

#### Syntax

```
mpd [ --help ] [ -V ] [ --version ] [ --host=<host> --port=<portnum> ] \  
    [ --noconsole ] [ --trace ] [ --echo ] [ --daemon ] [ --bulletproof ] \  
    [ --i fhn <interface/hostname> ] [ --listenport <listenport> ]
```

#### Arguments

<code>--help</code>	Display a help message
<code>-V   --version</code>	Display the Intel® MPI Library version information
<code>-h &lt;host&gt; -p &lt;portnum&gt;   --host=&lt;host&gt; --port= &lt;portnum&gt;</code>	Specify the host and port to be used for entering an existing ring. The <code>--host</code> and <code>--port</code> options must be specified together
<code>-n   --noconsole</code>	Do not create a console at startup
<code>-t   --trace</code>	Print internal MPD trace information

<code>-e   --echo</code>	Print a port number at startup to which other <code>mpds</code> may connect
<code>-d   --daemon</code>	Start <code>mpd</code> in daemon mode. By default, the interactive mode is enabled
<code>--bulletproof</code>	Turn MPD bulletproofing on
<code>--ifhn=&lt;interface/ hostname&gt;</code>	Specify <code>&lt;interface/hostname&gt;</code> to use for MPD communications
<code>-l &lt;listenport&gt;   --listenport= &lt;listenport&gt;</code>	Specify the <code>mpd</code> listening port

### Description

Multipurpose daemon\* (MPD) is the Intel® MPI Library process management system for starting parallel jobs. Before running a job, start `mpd` daemons on each host and connect them into a ring. Long parameter names may be abbreviated to their first letters by using only one hyphen and no equal sign. For example,

```
$ mpd -h masterhost -p 4268 -n
```

is equivalent to

```
$ mpd --host=masterhost --port=4268 -noconsole
```

If a file named `.mpd.conf` is presented in the user's home directory, only the user can have read and write privileges. The file must minimally contain a line with `secretword=<secretword>`. Create the `mpd.conf` file in the `/etc` directory instead of `.mpd.conf` in the root's home directory to run `mpd` as root. We do not recommend starting the MPD ring under the root account.

### mpdboot

Start `mpd` ring.

#### Syntax

```
mpdboot [ -h ] [ -V ] [ -n <#nodes> ] [ -f <hostsfile> ] [ -r <rshcmd> ] \  
        [ -u <user> ] [ -m <mpdcmd> ] [ --locons ] [ --remcons ] \  
        [ -s ] [ -d ] [ -v ] [ -l ] [ --ncpus=<ncpus> ] [ -o ] \  
        [ -b <maxbranch> ] [ -p ]
```

or

```
mpdboot [ --help ] [ --version ] [ --totalnum=<#nodes> ] \  
        [ --file=<hostsfile> ] [ --rsh=<rshcmd> ] [ --user=<user> ] \  
        [ --mpd=<mpdcmd> ] [ --locons ] [ --remcons ] [ --shell ] \  
        [ --debug ] [ --verbose ] [ -l ] [ --ncpus=<ncpus> ] [ --ordered ] \  
        [ --maxbranch=<maxbranch> ] [ --parallel-startup ]
```

#### Arguments

<code>-h   --help</code>	Display a help message
<code>-V   --version</code>	Display Intel® MPI Library version information
<code>-d   --debug</code>	Print debug information
<code>-v   --verbose</code>	Print extra verbose information. Show the <code>&lt;rshcmd&gt;</code> attempts

<code>-n &lt;#nodes&gt;   --totalnum=&lt;#nodes&gt;</code>	Number of nodes in <code>mpd.hosts</code> on which daemons are started
<code>-r &lt;rshcmd&gt;   --rsh=&lt;rshcmd&gt;</code>	Specify remote shell to start daemons and jobs. The default value is <code>ssh</code>
<code>-f &lt;hostsfile&gt;   --file=&lt;hostsfile&gt;</code>	Path/name of the file that has the list of machine names on which the daemons are started
<code>-l</code>	Remove the restriction of starting only one <code>mpd</code> per machine
<code>-m &lt;mpdcmd&gt;   --mpd=&lt;mpdcmd&gt;</code>	Specify the full path name of the <code>mpd</code> on the remote hosts
<code>-s   --shell</code>	Specify the shell
<code>-u &lt;user&gt;   -- user=&lt;user&gt;</code>	Specify the user
<code>--locons</code>	Do not create local MPD consoles
<code>--remcons</code>	Do not create remote MPD consoles
<code>--ncpus=&lt;ncpus&gt;</code>	Indicate how many processors to use on the local machine (other nodes are listed in the hosts file)
<code>-o   --ordered</code>	Start all the <code>mpd</code> daemons in the exact order as specified in the <code>mpd.hosts</code> file
<code>-b &lt;maxbranch&gt;   -- maxbranch=&lt;maxbranch&gt;</code>	Use this option to indicate the maximum number of the <code>mpd</code> daemons to enter the <code>mpd</code> ring under another. This helps to control the parallelism of the <code>mpd</code> ring start. The default value is four
<code>-p   --parallel- startup</code>	Use this option to allow parallel fast starting of <code>mpd</code> daemons under one local root. No daemon checking is performed. This option also supports shells which do not transfer the output from the remote commands

## Description

Start the `mpd` daemons on the specified number of nodes by providing a list of node names in `<mpd.hosts>`.

The `mpd` daemons are started using the `ssh` command by default. If the `ssh` connectivity is not enabled, use the `-r rsh` option to switch over to `rsh`. Make sure that all nodes in the cluster can connect to each other through the `ssh` command without a password or, if the `-r rsh` option is used, through the `rsh` command without a password.

**NOTE:** The `mpdboot` command spawns an MPD daemon on the host machine, even if the machine name is not listed in the `mpd.hosts` file.

## mpdexit

Shut down a single `mpd` daemon.

### Syntax

```
mpdexit [ --help ] [ -V ] [--version] <mpdid>
```

**Arguments**

<code>--help</code>	Display a help message
<code>-V   --version</code>	Display Intel® MPI Library version information
<code>&lt;mpdid&gt;</code>	Specify the <code>mpd</code> daemon to kill

**Description**

Use this command to cause the single `mpd` daemon to exit. Use `<mpdid>` obtained through the `mpdtrace -l` command in the form `<hostname>_<port number>`.

**mpdallexit**

Shut down all `mpd` daemons on all nodes.

**Syntax**

```
mpdallexit [ --help ] [ -V ] [ --version ]
```

**Arguments**

<code>--help</code>	Display a help message
<code>-V   --version</code>	Display Intel® MPI Library version information

**Description**

Use this command to shutdown all MPD rings you own.

**mpdcleanup**

Clean up the environment after an `mpd` crash.

**Syntax**

```
mpdcleanup [ -h ] [ -V ] [ -f <hostsfile> ] [ -r <rshcmd> ] [ -u <user> ] \
           [ -c <cleancmd> ] [ -a]
```

or

```
mpdcleanup [ --help ] [ --version ] [ --file=<hostsfile> ] \
           [ --rsh=<rshcmd> ] [ --user=<user> ] [ --clean=<cleancmd> ] \
           [ --all]
```

**Arguments**

<code>-h   --help</code>	Display a help message
<code>-V   --version</code>	Display Intel® MPI Library version information
<code>-f &lt;hostsfile&gt;   --file=&lt;hostsfile&gt;</code>	Specify the file containing a list of machines to clean up
<code>-r &lt;rshcmd&gt; / --rsh=&lt;rshcmd&gt;</code>	Specify the remote shell to use
<code>-u &lt;user&gt;   --user=&lt;user&gt;</code>	Specify the user

<code>-c &lt;cleancmd&gt;   --clean=&lt;cleancmd&gt;</code>	Specify the command to use for removing the UNIX* socket. The default command is <code>/bin/rm -f</code>
<code>-a   --all</code>	Kill all <code>mpd</code> daemons related to the current settings of the <code>I_MPI_JOB_CONTEXT</code> environment variable on all hosts specified in <code>&lt;hostsfile&gt;</code>

### Description

Use this command to clean up the environment after an `mpd` crash. It removes the UNIX\* socket on local and remote machines or kills all `mpd` daemons related to the current environment controlled by the `I_MPI_JOB_CONTEXT` environment variable.

For instance, use the following command to remove the UNIX sockets on machines specified in the `hostsfile` file:

```
$ mpidcleanup --file=hostsfile
```

Use the following command to kill the `mpd` daemons on the machines specified in the `hostsfile` file:

```
$ mpidcleanup --file=hostsfile --all
```

## mpdtrace

Determine whether `mpd` is running.

### Syntax

```
mpdtrace [ --help ] [ -V ] [ --version ] [ -l ]
```

### Arguments

<code>--help</code>	Display a help message
<code>-V   --version</code>	Display Intel® MPI Library version information
<code>-l</code>	Show MPD identifiers instead of the hostnames

### Description

Use this command to list the hostnames or identifiers of all `mpds` in the ring. The output identifiers have the form `<hostname>_<port number>`.

## mpdcheck

Check for configuration problems on the host or print configuration information about this host.

### Syntax

```
mpdcheck [ -v ] [ -l ] [ -h ] [ --help ] [ -V ] [ --version ]
```

```
mpdcheck -pc [ -v ] [ -l ]
```

```
mpdcheck -f <host_file> [ -ssh ] [ -v ] [ -l ]
```

```
mpdcheck -s [ -v ] [ -l ]
```

```
mpdcheck -c <server_host> <server_port> [ -v ] [ -l ]
```

### Arguments

<code>-h   --help</code>	Display a help message
<code>-V   --version</code>	Display Intel® MPI Library version information
<code>-pc</code>	Print configuration information about a local host
<code>-f &lt;host_file&gt;</code>	Print information about the hosts listed in <code>&lt;host_file&gt;</code>

<code>-ssh</code>	Invoke testing of <code>ssh</code> on each remote host. Use in conjunction with the <code>-f</code> option
<code>-s</code>	Run <code>mpdcheck</code> as a server on one host
<code>-c &lt;server_host&gt; &lt;server_port&gt;</code>	Run <code>mpdcheck</code> as a client on the current or different host. Connect to the <code>&lt;server_host&gt; &lt;server_port&gt;</code>
<code>-l</code>	Print diagnostic messages in extended format
<code>-v</code>	Print the actions that <code>mpdcheck</code> is performing

### Description

Use this command to check configuration problems on the cluster nodes. Any output started with `***` indicates a potential problem.

If you have problems running parallel jobs through `mpd` on one or more hosts, try to run the script once on each of those hosts.

## mpdringtest

Test the MPD ring.

### Syntax

```
mpdringtest [ --help ] [ -V ] [ --version ] <number of loops>
```

### Arguments

<code>--help</code>	Display a help message
<code>-V   --version</code>	Display Intel® MPI Library version information
<code>&lt;number of loops&gt;</code>	Number of loops

### Description

Use this command to test how long it takes for a message to circle the `mpd` ring.

## mpdlistjobs

List the running processes for a particular set of MPI jobs.

### Syntax

```
mpdlistjobs [ -h ] [ -V ] [ -u <username> ] [ -a <jobalias> ] [ -j <jobid> ]
```

or

```
mpdlistjobs [ --help ] [ --version ] [ --user=<username> ] \  
[ --alias=<jobalias> ] [ --jobid=<jobid> ]
```

### Arguments

<code>-h   --help</code>	Display a help message
<code>-V   --version</code>	Display Intel® MPI Library version information
<code>-u &lt;username&gt;   --user=&lt;username&gt;</code>	List jobs of a particular user
<code>-a &lt;jobalias&gt;   --alias=&lt;jobalias&gt;</code>	List information about the particular job specified by <code>&lt;jobalias&gt;</code>
<code>-j &lt;jobid&gt;  </code>	List information about the particular job specified by <code>&lt;jobid&gt;</code>

<code>--jobid=&lt;jobid&gt;</code>	
------------------------------------	--

### Description

Use this command to list the running processes for a set of MPI jobs. All jobs for the current machine are displayed by default.

## mpdsigjob

Apply a signal to a process running an application.

### Syntax

```
mpdsigjob [ --help ] [ -V ] [ --version ] <sigtype> \
          [-j <jobid> | -a <jobalias> ] [-s | -g ]
```

### Arguments

<code>--help</code>	Display a help message
<code>-V   --version</code>	Display Intel® MPI Library version information
<code>&lt;sigtype&gt;</code>	Specify the signal to send
<code>-a &lt;jobalias&gt;</code>	Send a signal to the job specified by <code>&lt;jobalias&gt;</code>
<code>-j &lt;jobid&gt;</code>	Send a signal to the job specified by <code>&lt;jobid&gt;</code>
<code>-s</code>	Deliver a signal to a single user process
<code>-g</code>	Deliver a signal to a group of processes. This is the default behavior

### Description

Use this command to deliver a specific signal to the processes of a running job. The desired signal is the first argument. Specify only one of two options: `-j` or `-a`.

## mpdkilljob

Kill a job.

### Syntax

```
mpdkilljob [ --help ] [ -V ] [ --version ] [ <jobnum> ] [ -a <jobalias> ]
```

### Arguments

<code>--help</code>	Display a help message
<code>-V   --version</code>	Display Intel® MPI Library version information
<code>&lt;jobnum&gt;</code>	Kill the job specified by <code>&lt;jobnum&gt;</code>
<code>-a &lt;jobalias&gt;</code>	Kill the job specified by <code>&lt;jobalias&gt;</code>

### Description

Use this command to kill the job specified by `<jobnum>` or by `<jobalias>`. Obtain `<jobnum>` and `<jobalias>` from the `mpdlistjobs` command. The `<jobid>` field has the following format: `<jobnum>@<mpdid>`.

## mpdhelp

Print brief help concerning MPD commands.

Syntax

```
mpdhelp [ -V ] [ --version ]
```

Arguments

-V   --version	Display Intel® MPI Library version information
----------------	--

Description

Use this command to obtain a brief help message concerning MPD commands.

2.4.1 Job Startup Commands

mpiexec

Syntax

```
mpiexec <g-options> <l-options> <executable>
```

or

```
mpiexec <g-options> <l-options> <executable1> : \  
<l-options> <executable2>
```

or

```
mpiexec -configfile <file>
```

Arguments

<g-options>	Global options that apply to all MPI processes
<l-options>	Local options that apply to a single arg-set
<executable>	./a.out or path/name of the executable file
<file>	File with command-line options

Description

By using the first command-line syntax, you can start all MPI processes of the <executable> with the single arg-set. For example, the following command executes a.out over the specified <# of processes>:

```
$ mpiexec -n <# of processes> ./a.out
```

By using the second command-line syntax, you can start several MPI programs or the same MPI program with different arg-sets. For example, the following command would run each given executable on a different host:

```
$ mpiexec -n 2 -host host1 ./a.out : \  
-n 2 -host host2 ./b.out
```

In the third command-line syntax, read the command line from specified <file>. For a command with a single arg-set, the entire command should be specified on a single line in <file>. For a command with multiple arg-sets, each arg-set should be specified on a single, separate line in <file>. Global options should always appear at the beginning of the first line in <file>.

MPD daemons must already be running in order for mpiexec to succeed.

**NOTE:** If there is no "." in the `PATH` environment variable on all nodes in the cluster, specify `<executable>` as `./a.out` rather than `a.out`.

### 2.4.1.1 Extended Device Control Options

Use these options to select a specific fabric combination.

The exact combination of fabrics depends on the number of processes started per node.

If all processes start on one node, the Intel® MPI library uses `shm` intra-node communication regardless of the selected option from the list in this topic.

If the number of started processes is less than or equal to the number of available nodes, the library uses the first available fabric from the list of fabrics for inter-nodes communication.

For other cases, the library uses `shm` for intra-node communication, and the first available fabric from the list of fabrics for inter-nodes communication. See [I\\_MPI\\_FABRICS](#) and [I\\_MPI\\_FABRICS\\_LIST](#) for more details.

The `shm` fabric is available for both Intel® and non-Intel microprocessors, but it may perform additional optimizations for Intel microprocessors than it performs for non-Intel microprocessors.

#### -rdma

Use this option to select an RDMA-capable network fabric for inter-nodes communication. The application attempts to use first available RDMA-capable network fabric from the list `dapl` or `ofa`. If no such fabric is available, other fabrics from the list `tcp` or `tmi` are used. This option is equivalent to the `-genv I_MPI_FABRICS_LIST dapl,ofa,tcp,tmi -genv I_MPI_FALLBACK 1` setting.

#### -RDMA

Use this option to select an RDMA-capable network fabric for inter-nodes communication. The application attempts to use first available RDMA-capable network fabric from the list `dapl` or `ofa`. The application fails if no such fabric is found. This option is equivalent to the `-genv I_MPI_FABRICS_LIST dapl,ofa -genv I_MPI_FALLBACK 1` setting.

#### -dapl

Use this option to select DAPL capable network fabric for inter-nodes communication. The application attempts to use DAPL capable network fabric. If no such fabric is available, another fabrics from the list `tcp`, `tmi` or `ofa` is used. This option is equivalent to the `-genv I_MPI_FABRICS_LIST dapl,tcp,tmi,ofa -genv I_MPI_FALLBACK 1` setting.

#### -DAPL

Use this option to select DAPL capable network fabric for inter-nodes communication. The application fails if no such fabric is found. This option is equivalent to the `-genv I_MPI_FABRICS_LIST dapl -genv I_MPI_FALLBACK 0` setting.

#### -ib

Use this option to select OFA capable network fabric for inter-nodes communication. The application attempts to use OFA capable network fabric. If no such fabric is available, another fabrics from the list `dapl`, `tcp` or `tmi` is used. This option is equivalent to the `-genv I_MPI_FABRICS_LIST ofa,dapl,tcp,tmi -genv I_MPI_FALLBACK 1` setting.

#### -IB

Use this option to select OFA capable network fabric for inter-nodes communication. The application fails if no such fabric is found. This option is equivalent to the `-genv I_MPI_FABRICS_LIST ofa -genv I_MPI_FALLBACK 0` setting.

#### -tmi

Use this option to select TMI capable network fabric for inter-nodes communication. The application attempts to use TMI capable network fabric. If no such fabric is available, another fabrics from the list

`dapl,tcp` or `ofa` is used. This option is equivalent to the `-genv I_MPI_FABRICS_LIST tmi,dapl,tcp,ofa -genv I_MPI_FALLBACK 1` setting.

### -TMI

Use this option to select TMI capable network fabric for inter-nodes communication. The application will fail if no such fabric is found. This option is equivalent to the `-genv I_MPI_FABRICS_LIST tmi -genv I_MPI_FALLBACK 0` setting.

### -mx

Use this option to select Myrinet MX\* network fabric for inter-nodes communication. The application attempts to use Myrinet MX\* network fabric. If no such fabric is available, another fabrics from the list `dapl,tcp` or `ofa` is used. This option is equivalent to the `-genv I_MPI_FABRICS_LIST tmi,dapl,tcp,ofa -genv I_MPI_TMI_PROVIDER mx -genv I_MPI_DAPL_PROVIDER mx -genv I_MPI_FALLBACK 1` setting.

### -MX

Use this option to select Myrinet MX\* network fabric for inter-nodes communication. The application fails if no such fabric is found. This option is equivalent to the `-genv I_MPI_FABRICS_LIST tmi -genv I_MPI_TMI_PROVIDER mx -genv I_MPI_FALLBACK 0` setting.

### -psm

Use this option to select Qlogic\* network fabric for inter-nodes communication. The application attempts to use Qlogic\* network fabric. If no such fabric is available, another fabrics from the list `dapl,tcp` or `ofa` is used. This option is equivalent to the `-genv I_MPI_FABRICS_LIST tmi,dapl,tcp,ofa -genv I_MPI_TMI_PROVIDER psm -genv I_MPI_FALLBACK 1` setting.

### -PSM

Use this option to select Qlogic\* network fabric for inter-nodes communication. The application fails if no such fabric is found. This option is equivalent to the `-genv I_MPI_FABRICS_LIST tmi -genv I_MPI_TMI_PROVIDER psm -genv I_MPI_FALLBACK 0` setting.

### -gm

Use this option to select Myrinet\* GM\* network fabric for inter-nodes communication. This option is equivalent to the `-genv I_MPI_DEVICE rdssm:GmHca0 -genv I_MPI_FALLBACK_DEVICE 1` setting.

**NOTE:** This environment variable is deprecated and supported mostly for backward compatibility.

### -GM

Use this option to select Myrinet\* GM\* network fabric for inter-nodes communication. The application fails if no such fabric is found. This option is equivalent to the `-genv I_MPI_DEVICE rdssm:GmHca0 -genv I_MPI_FALLBACK_DEVICE 0` setting.

**NOTE:** This environment variable is deprecated and supported mostly for backward compatibility.

## 2.4.1.2 Global Options

### -version or -V

Use this option to display Intel® MPI Library version information.

### -h or -help or --help

Use this option to display the `mpiexec` help message.

### -tune [ <arg >]

where:

`<arg> = {<dir_name>, <configuration_file>}.`

Use this option to optimize the Intel® MPI Library performance using data collected by the `mpitune` utility.

If `<arg>` is not specified, the best-fit tune options will be selected for the given configuration. The default location of the configuration file is `<installdir>/<arch>/etc` directory. You can override this default location by explicitly stating: `<arg>=<dir_name>`. The provided configuration file is used if you set `<arg>=<configuration_file>`.

See [Automatic Tuning Utility](#) for more details.

## **-nolocal**

Use this option to avoid running `<executable>` on the host where the `mpiexec` is launched. This option is useful, for example, on clusters that deploy a dedicated master node for starting the MPI jobs, and a set of compute nodes for running the actual MPI processes.

## **-perhost <# of processes>**

Use this option to place the indicated number of consecutive MPI processes on every host in group round robin fashion. The total number of processes to start is controlled by the `-n` option as usual.

The `mpiexec` command controls how the ranks of the processes are allocated to the nodes in the cluster. By default, `mpiexec` uses group round-robin assignment of ranks to nodes, putting consecutive MPI processes on all processor cores.

To change this default behavior, set the number of processes per host by using the `-perhost` option, and set the total number of processes by using the `-n` option. See [Local Options](#) for details. The first `<# of processes>` indicated by the `-perhost` option is executed on the first host; the next `<# of processes>` is executed on the next host, and so on.

See also the [I\\_MPI\\_PERHOST](#) environment variable.

## **-rr**

Use this option to place consecutive MPI processes onto different host in round robin fashion. This option is equivalent to `-perhost 1`.

## **-grr <# of processes>**

Use this option to place the indicated number of consecutive MPI processes on every host in group round robin fashion. This option is equivalent to `-perhost <# of processes>`.

## **-ppn <# of processes>**

Use this option to place the indicated number of consecutive MPI processes on every host in group round robin fashion. This option is equivalent to `-perhost <# of processes>`.

## **-machinefile <machine file>**

Use this option to control the process placement through `<machine file>`. The total number of processes to start is controlled by the `-n` option as usual.

A machine file is a list of fully qualified or short host names, one name per line. Blank lines and lines that start with `#` as the first character are ignored.

By repeating a host name, you place additional processes on this host. You can also use the following format to avoid repetition of the same host name: *<host name>:<number of processes>*. For example, the following machine files:

host1

host1

host2

host2

host3

is equivalent to:

host1:2

host2:2

host3

It is also possible to specify the network interface used for communication for each node: *<host name>:<number of processes> [ifhn=<interface\_host\_name>]*.

**NOTE:** The *-machinefile*, *-ppn*, *-rr*, and *-perhost* options are intended for process distribution. Do not use them simultaneously. Otherwise *-machinefile* takes precedence.

### *-g <l-option>*

Use this option to apply the named local option *<l-option>* globally. See [Local Options](#) for a list of all local options. During the application startup, the default value is the *-genvuser* option. The options *-genvnone*, *-genvuser*, *-genvall* have the lowest priority, *-genvlist*, *-genvexcl* have higher priority than the previous set. The *-genv* option has the highest priority. Local options have higher priority than the global options.

### *-genv <ENVVAR> <value>*

Use this option to set the *<ENVVAR>* environment variable to the specified *<value>* for all MPI processes.

### *-genvuser*

Use this option to propagate all user environment variables to all MPI processes, with the exception of the following system environment variables: *\$HOSTNAME*, *\$HOST*, *\$HOSTTYPE*, *\$MACHTYPE*, *\$OSTYPE*. This is the default setting.

### *-genvall*

Use this option to enable propagation of all environment variables to all MPI processes.

### *-genvnone*

Use this option to suppress propagation of any environment variables to any MPI processes.

### *(SDK only) -trace [<profiling\_library>] or -t [<profiling\_library>]*

Use this option to profile your MPI application using the indicated *<profiling\_library>*. If the *<profiling\_library>* is not mentioned, the default profiling library *libVT.so* is used.

Set the *I\_MPI\_JOB\_TRACE\_LIBS* environment variable to override the default profiling library.

**NOTE:** It is not necessary to link your application against the profiling library before execution.

**(SDK only) -check\_mpi [<checking\_library>]**

Use this option to check your MPI application using the indicated *<checking\_library>*. If *<checking\_library>* is not mentioned, the default checking library *libVTmc.so* is used.

Set the *I\_MPI\_JOB\_CHECK\_LIBS* environment variable to override the default checking library.

**NOTE:** It is not necessary to link your application against the checking library before execution.

**-tv**

Use this option to run *<executable>* under the TotalView\* debugger. For example:

```
$ mpiexec -tv -n <# of processes> <executable>
```

See [Environment Variables](#) for information on how to select the TotalView\* executable file.

**NOTE:** Ensure the environment variable *TVDSVRLAUNCHCMD=ssh* because the TotalView\* uses *rsh* by default.

**NOTE:** The TotalView\* debugger has a feature to displays the message queue state of your MPI program. To use the state display feature, do the following steps:

1. Run your *<executable>* with *-tv* option.

```
$ mpiexec -tv -n <# of processes> <executable>
```

2. Answer **Yes** to the question about stopping the Python\* job.

To display the internal state of the MPI library textually, select **Tools > Message Queue** command. If you select the **Process Window Tools > Message Queue Graph** command, the TotalView\* displays a window that shows a graph of the current message queue state. For more information, see [TotalView\\*](#).

**-tva <jobid>**

Use this option to attach the TotalView\* debugger to existing *<jobid>*. For example:

```
$ mpiexec -tva <jobid>
```

**-tvsu**

Use this option to run *<executable>* for later attachment with the TotalView\* debugger. For example:

```
$ mpiexec -tvsu -n <# of processes> <executable>
```

**NOTE:** To debug the running Intel® MPI job, attach the TotalView\* to the Python\* instance that is running the *mpiexec* script.

**-idb**

Use this option to run *<executable>* under the Intel® Debugger. For example:

```
$ mpiexec -idb -n <# of processes> <executable>
```

Include the installation path of the Intel® Debugger in the *IDB\_HOME* environment variable.

**-idba <jobid>**

Use this option to attach the Intel® Debugger to the existing *<jobid>*. For example:

```
$ mpiexec -idba <jobid>
```

**-gdb**

Use this option to run *<executable>* under the GNU\* debugger. For example:

```
$ mpiexec -gdb -n <# of processes> <executable>
```

**-gdba <jobid>**

Use this option to attach the GNU\* debugger to the existing *<jobid>*. For example:

```
$ mpiexec -gdba <jobid>
```

**-a <alias>**

Use this option to assign *<alias>* to the job.

**-ordered-output**

Use this option to avoid intermingling of data output by the MPI processes. This option affects both the standard output and standard error streams.

**NOTE:** For this option to work, the last line output by each process must end with the end-of-line ('\n') character. Otherwise the application may stop responding.

**-m**

Use this option to merge output lines.

**-l**

Use this option to insert the MPI process rank at the beginning of all lines written to the standard output.

**-s <spec>**

Use this option to direct standard input to the specified MPI processes.

**Arguments**

<i>&lt;spec&gt;</i>	Define MPI process ranks
<i>all</i>	Use all processes
<i>&lt;l&gt;, &lt;m&gt;, &lt;n&gt;</i>	Specify an exact list and use processes <i>&lt;l&gt;</i> , <i>&lt;m&gt;</i> and <i>&lt;n&gt;</i> only. The default value is zero
<i>&lt;k&gt;, &lt;l&gt;-&lt;m&gt;, &lt;n&gt;</i>	Specify a range and use processes <i>&lt;k&gt;</i> , <i>&lt;l&gt;</i> through <i>&lt;m&gt;</i> , and <i>&lt;n&gt;</i>

**-noconf**

Use this option to disable processing of the *mpiexec* configuration files described in the section [Configuration Files](#).

**-ifhn <interface/hostname>**

Use this option to specify the network interface for communication with the local MPD daemon. The *<interface/hostname>* should be an IP address or a hostname associated with the alternative network interface.

**-ecfn <filename>**

Use this option to output XML exit codes to the file *<filename>*.

**-configfile <filename>**

Use this option to specify the file *<filename>* that contains command-line options. Blank lines and lines that start with # as the first character are ignored. For example, the configuration file contains the following commands to run the executables *a.out* and *b.out* using the *rdssm* device over *host1* and *host2* respectively:

```
-host host1 -env I_MPI_DEBUG 2 -env I_MPI_DEVICE rdssm -n 2 ./a.out
-host host2 -env I_MPI_DEBUG 2 -env I_MPI_DEVICE rdssm -n 2 ./b.out
```

To launch a MPI application according to the parameters above, use:

```
$ mpiexec -configfile <filename>
```

**NOTE:** This option may only be used alone. It terminates parsing of the *mpiexec* command line.

### 2.4.1.3 Local Options

**-n <# of processes> or -np <# of processes>**

Use this option to set the number of MPI processes to run with the current arg-set.

**-env <ENVVAR> <value>**

Use this option to set the *<ENVVAR>* environment variable to specified *<value>* for all MPI processes in the current arg-set.

**-envuser**

Use this option to propagate all user environment variables with the exception of the following variables: *\$HOSTNAME*, *\$HOST*, *\$HOSTTYPE*, *\$MACHTYPE*, *\$OSTYPE*. This is the default setting.

**-envall**

Use this option to propagate all environment variables in the current environment.

**-envnone**

Use this option to suppress propagation of any environment variables to the MPI processes in the current arg-set.

**-envlist <list of env var names>**

Use this option to pass a list of environment variables with their current values. *<list of env var names>* is a comma separated list of environment variables to be sent to the processes. If this option is used several times in the command line, all variables listed in the arguments are included into one list.

**-envexcl <list of env var names>**

Use this option to suppress propagation of the listed environment variables to the MPI processes in the current arg-set.

**-host <nodename>**

Use this option to specify a particular <nodename> on which the MPI processes in the current arg-set are to be run. For example, the following command runs the executable `a.out` on host `host1` only:

```
$ mpiexec -n 2 -host host1 ./a.out
```

**-path <directory>**

Use this option to specify the path to <executable> that is to be run in the current arg-set.

**-wdir <directory>**

Use this option to specify the working directory in which <executable> is to be run in the current arg-set.

**-umask <umask>**

Use this option to perform the `umask <umask>` command for the remote process.

## 2.4.1.4 Configuration Files

The `mpiexec` configuration files specify the default options applied to all `mpiexec` commands.

If any of these files exist, their contents are prefixed to the command-line options for `mpiexec` in the following order:

1. System-wide `<installdir>/etc/mpiexec.conf`. The default location of the configuration file is the `<installdir>/<arch>/etc`.
2. User-specific `$HOME/.mpiexec.conf`
3. Session-specific `$PWD/mpiexec.conf`

You can override these files by defining environment variables and using command line options. You can skip these configuration files by using the `mpiexec -noconf` option.

You can create or modify these files. They contain `mpiexec` command-line options. Blank lines and lines that start with `#` are ignored. For example, to specify a default device, add the following line to the respective `mpiexec.conf` file:

```
-genv I_MPI_DEVICE <device>
```

## 2.4.1.5 Environment Variables

**I\_MPI\_DEBUG**

Print out debugging information when an MPI program starts running.

**Syntax**

```
I_MPI_DEBUG=<level>[, <flags>]
```

**Arguments**

<level>	Indicate level of debug information provided
0	Output no debugging information. This is the default value
1	Output verbose error diagnostics
2	Confirm which <code>I_MPI_FABRICS</code> ( <code>I_MPI_DEVICE</code> ) was used
3	Output effective MPI rank, <code>pid</code> and node mapping table

4	Output process pinning information
5	Output Intel MPI-specific environment variables
> 5	Add extra levels of debug information

<flags>	Comma-separated list of debug flags
pid	Show process id for each debug message
tid	Show thread id for each debug message for multithreaded library
time	Show time for each debug message
datetime	Show time and date for each debug message
host	Show host name for each debug message
level	Show level for each debug message
scope	Show scope for each debug message
line	Show source line number for each debug message
file	Show source file name for each debug message
nofunc	Do not show routine name
norank	Do not show rank
flock	Synchronize debug output from different process or threads
nobuf	Do not use buffered I/O for debug output

### Description

Set this environment variable to control the output of the debugging information.

You can specify the output file name for debug information by setting the `I_MPI_DEBUG_OUTPUT` environment variable.

Each printed line has the following format:

```
[<identifier>] <message>
```

where *<identifier>* identifies the MPI process that produced the message, while *<message>* contains the debugging output.

The *<identifier>* is an MPI process rank if *<level>* is an unsigned number. If the '+' sign is added in front of the *<level>* number, the *<identifier>* contains a `rank#pid@hostname` tuple. Here, `rank` is the MPI process rank, `pid` is the UNIX process id, and `hostname` is the host name as defined at process launch time.

For example, the following command:

```
$ mpiexec -n 1 -env I_MPI_DEBUG 2 ./a.out
```

may produce the following output:

```
[0] MPI startup(): shared memory data transfer mode
```

while the command

```
$ mpiexec -n 1 -env I_MPI_DEBUG +2 ./a.out
```

or

```
$ mpiexec -n 1 -env I_MPI_DEBUG 2,pid,host ./a.out
```

may produce the following output:

```
[0#1986@mpiclust001] MPI startup(): shared memory data transfer mode
```

**NOTE:** Compiling with `mpiicc -g` causes considerable amount of additional debug information to be printed.

I\_MPI\_DEBUG\_OUTPUT

Set output file name for debug information.

Syntax

```
I_MPI_DEBUG_OUTPUT =<arg>
```

Arguments

<arg>	String value
stdout	Output to stdout - default value
stderr	Output to stderr
<file_name>	Specify the output file name for debug information

Description

Set this environment variable if you want to split output of debug information from the output produced by an application. If you use format like `'%r'`, `'%p'` or `'%h'`, rank, pid or host name is added to the file name accordingly.

I\_MPI\_PERHOST

Define the default settings for the `-perhost` option in the `mpiexec` command.

Syntax

```
I_MPI_PERHOST=<value>
```

Arguments

<value>	Define the default process layout
<n> > 0	<n> processes per node
all	All logical CPUs on a node
allcores	All cores (physical CPUs) on a node

Description

Set this environment variable to define the default setting for the `-perhost` option. If `-perhost` is explicitly called in the command line, the `I_MPI_PERHOST` environment variable has no effect. The `-perhost` option assumes the value of the `I_MPI_PERHOST` environment variable if this environment variable is defined.

**NOTE:** `I_MPI_PERHOST` is incompatible with the `mpiexec -host` option. The `I_MPI_PERHOST` environment variable is ignored in this case.

## I\_MPI\_PRINT\_VERSION

Print library version information.

### Syntax

`I_MPI_PRINT_VERSION=<arg>`

### Arguments

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Print library version information.
<code>disable   no   off   0</code>	No action. This is the default value.

### Description

Set this environment variable to enable/disable printing of Intel® MPI library version information when an MPI application starts running.

## (SDK only) I\_MPI\_JOB\_TRACE\_LIBS

### (MPIEXEC\_TRACE\_LIBS)

Choose the libraries to preload through the `-trace` option.

### Syntax

`I_MPI_JOB_TRACE_LIBS=<arg>`

### Deprecated Syntax

`MPIEXEC_TRACE_LIBS=<arg>`

### Arguments

<code>&lt;arg&gt;</code>	String parameter
<code>&lt;list&gt;</code>	Blank separated list of libraries to preload. The default value is <code>vt</code>

### Description

Set this environment variable to choose an alternative library for preloading by the `-trace` option.

## (SDK only) I\_MPI\_JOB\_CHECK\_LIBS

Choose the libraries to preload through the `-check_mpi` option.

### Syntax

`I_MPI_JOB_CHECK_LIBS=<arg>`

### Arguments

<code>&lt;arg&gt;</code>	String parameter
<code>&lt;list&gt;</code>	Blank separated list of libraries to preload. The default value is <code>vtmc</code>

### Description

Set this environment variable to choose an alternative library for preloading by the `-check_mpi` option.

I\_MPI\_JOB\_STARTUP\_TIMEOUT

Set the `mpiexec` job startup timeout.

Syntax

`I_MPI_JOB_STARTUP_TIMEOUT=<timeout>`

Arguments

<code>&lt;timeout&gt;</code>	Define <code>mpiexec</code> job startup timeout period in seconds
<code>&lt;n&gt; &gt;= 0</code>	The default timeout value is 20 seconds

Description

Set this environment variable to make `mpiexec` wait for the job to start in `<timeout>` seconds after its launch. The `<timeout>` value should be greater than zero. Otherwise the environment variable setting is ignored and a warning message is printed. Setting this environment variable may make sense on large clusters with a lot of nodes where the job startup time may exceed the default value.

**NOTE:** Set the `I_MPI_JOB_STARTUP_TIMEOUT` environment variable in the shell environment before executing the `mpiexec` command. Do not use the `-genv` or `-env` options for setting the `<timeout>` value. Those options are used only for passing environment variables to the MPI process environment.

I\_MPI\_JOB\_TIMEOUT

(MPIEXEC\_TIMEOUT)

Set the `mpiexec` timeout.

Syntax

`I_MPI_JOB_TIMEOUT=<timeout>`

Deprecated Syntax

`MPIEXEC_TIMEOUT=<timeout>`

Arguments

<code>&lt;timeout&gt;</code>	Define <code>mpiexec</code> timeout period in seconds
<code>&lt;n&gt; &gt;= 0</code>	The default timeout value is zero, meaning no timeout

Description

Set this environment variable to make `mpiexec` terminate the job in `<timeout>` seconds after its launch. The `<timeout>` value should be greater than zero. Otherwise the environment variable setting is ignored.

**NOTE:** Set the `I_MPI_JOB_TIMEOUT` environment variable in the shell environment before executing the `mpiexec` command. Do not use the `-genv` or `-env` options for setting the `<timeout>` value. Those options are used only for passing environment variables to the MPI process environment.

**I\_MPI\_JOB\_TIMEOUT\_SIGNAL**

**(MPIEXEC\_TIMEOUT\_SIGNAL)**

Define a signal to be used when a job is terminated because of a timeout.

**Syntax**

I\_MPI\_JOB\_TIMEOUT\_SIGNAL=<number>

**Deprecated Syntax**

MPIEXEC\_TIMEOUT\_SIGNAL= <number>

**Arguments**

<number>	Define signal number
<n> > 0	The default value is 9 (SIGKILL)

**Description**

Define a signal number for task termination upon the timeout period specified by the environment variable I\_MPI\_JOB\_TIMEOUT. If you set a signal number unsupported by the system,, mpiexec prints a warning message and continues task termination using the default signal number 9 (SIGKILL).

**I\_MPI\_JOB\_ABORT\_SIGNAL**

Define a signal to be sent to all processes when a job is terminated unexpectedly.

**Syntax**

I\_MPI\_JOB\_ABORT\_SIGNAL=<number>

**Arguments**

<number>	Define signal number
<n> > 0	The default value is 9 (SIGKILL)

**Description**

Set this environment variable to define a signal for task termination. If you set an unsupported signal number, mpiexec prints a warning message and uses the default signal 9 (SIGKILL).

**I\_MPI\_JOB\_SIGNAL\_PROPAGATION**

**(MPIEXEC\_SIGNAL\_PROPAGATION)**

Control signal propagation.

**Syntax**

I\_MPI\_JOB\_SIGNAL\_PROPAGATION=<arg>

**Deprecated Syntax**

MPIEXEC\_SIGNAL\_PROPAGATION=<arg>

**Arguments**

<arg>	Binary indicator
enable   yes   on   1	Turn on propagation.

<code>disable   no   off   0</code>	Turn off propagation. This is the default value
-------------------------------------	---

Description

Set this environment variable to control propagation of the signals (`SIGINT`, `SIGALRM`, and `SIGTERM`) that may be received by the MPD daemons. If signal propagation is enabled, the received signal is sent to all processes of the MPI job. If signal propagation is disabled, all processes of the MPI job are stopped with the default signal 9 (`SIGKILL`).

I\_MPI\_OUTPUT\_CHUNK\_SIZE

Set the size of the `stdout/stderr` output buffer.

Syntax

`I_MPI_OUTPUT_CHUNK_SIZE=<size>`

Arguments

<code>&lt;size&gt;</code>	Define output chunk size in kilobytes
<code>&lt;n&gt; &gt; 0</code>	The default chunk size value is 1 KB

Description

Set this environment variable to increase the size of the buffer used to intercept the standard output and standard error streams from the processes. If the `<size>` value is not greater than zero, the environment variable setting is ignored and a warning message is displayed.

Use this setting for applications that create significant amount of output from different processes. With the `-ordered-output mpiexec` option, this setting helps to prevent the output from garbling.

**NOTE:** Set the `I_MPI_OUTPUT_CHUNK_SIZE` environment variable in the shell environment before executing the `mpiexec` command. Do not use the `-genv` or `-env` options for setting the `<size>` value. Those options are used only for passing environment variables to the MPI process environment.

I\_MPI\_PMI\_EXTENSIONS

Turn on/off the use of the Intel® MPI Library Process Management Interface (PMI) extensions.

Syntax

`I_MPI_PMI_EXTENSIONS=<arg>`

Arguments

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Turn on the PMI extensions
<code>disable   no   off   0</code>	Turn off the PMI extensions

Description

The Intel® MPI Library automatically detects if your process manager supports the PMI extensions. If supported, the extensions substantially decrease task startup time. Set `I_MPI_PMI_EXTENSIONS` to `disable` if your process manager does not support these extensions.

I\_MPI\_JOB\_FAST\_STARTUP

(I\_MPI\_PMI\_FAST\_STARTUP)

Turn on/off the faster Intel® MPI Library process startup algorithm.

Syntax

I\_MPI\_JOB\_FAST\_STARTUP=<arg>

Deprecated Syntax

I\_MPI\_PMI\_FAST\_STARTUP= <arg>

Arguments

<arg>	Binary indicator
enable   yes   on   1	Turn on the algorithm for fast startup. This is the default value
disable   no   off   0	Turn off the algorithm for fast startup

Description

The new algorithm significantly decreases the application startup time. Some DAPL providers may be overloaded during startup of large number of processes (greater than 512). To avoid this problem, turn off this algorithm by setting the I\_MPI\_JOB\_FAST\_STARTUP environment variable to disable.

TOTALVIEW\*

Select a particular TotalView\* executable file to use.

Syntax

TOTALVIEW=<path>

Arguments

<path>	Path/name of the TotalView* executable file instead of the default totalview
--------	--

Description

Set this environment variable to select a particular TotalView\* executable file.

IDB\_HOME

Set the Intel® Debugger installation directory path.

Syntax

IDB\_HOME=<path>

Arguments

<path>	Specify the installation directory of the Intel® Debugger
--------	---

Description

Set this environment variable to specify the installation directory of the Intel® Debugger.

## 2.4.2 Configuration Files

### \$HOME/.mpd.conf

This optional configuration file contains an `mpd` daemon password. Create it before setting up the `mpd` daemons. Use it to control access to the daemons by various Intel® MPI Library users.

#### Syntax

The file has a single line:

```
secretword=<mpd password>
```

or

```
MPD_SECRETWORD=<mpd password>
```

#### Description

An arbitrary `<mpd password>` string only controls access to the `mpd` daemons by various cluster users. Do not use Linux\* OS login passwords here.

Place the `$HOME/.mpd.conf` file on a network-mounted file system, or replicate this file so that it is accessible as `$HOME/.mpd.conf` on all nodes of the cluster.

When `mpdboot` is executed by some non-root `<user>`, this file should have user and ownership set to `<user>` and `<<user>'s group>` accordingly. The access permissions should be set to `600` mode (only user has read and write privileges).

**NOTE:** `MPD_SECRETWORD` is a synonym for `secretword`.

### mpd.hosts

This file has a list of node names which the `mpdboot` command uses to start `mpd` daemons.

Ensure that this file is accessible by the user who runs `mpdboot` on the node where the `mpdboot` command is actually invoked.

#### Syntax

The format of the `mpd.hosts` file is a list of node names, one name per line. Blank lines and the portions of any lines that follow a `#` character are ignored.

## 2.4.3 Environment Variables

### I\_MPI\_JOB\_CONFIG\_FILE

#### (I\_MPI\_MPD\_CONF)

Set the path/name of the `mpd` configuration file.

#### Syntax

```
I_MPI_JOB_CONFIG_FILE=<path/name>
```

#### Deprecated Syntax

```
I_MPI_MPD_CONF=<path/name>
```

#### Arguments

<code>&lt;path/name&gt;</code>	Absolute path of the MPD configuration file
--------------------------------	---

Description

Set this environment variable to define the absolute path of the file that is used by the `mpdboot` script instead of the default value `${HOME}/.mpd.conf`.

I\_MPI\_JOB\_CONTEXT

(MPD\_CON\_EXT)

Set a unique name for the `mpd` console file. This enables you to run several `mpd` rings under the same user account.

Syntax

I\_MPI\_JOB\_CONTEXT=<tag>

Deprecated Syntax

MPD\_CON\_EXT=<tag>

Arguments

<tag>	Unique MPD identifier
-------	-----------------------

Description

Set this environment variable to different unique values to allow several MPD rings to co-exist. Each MPD ring is associated with a separate `I_MPI_JOB_CONTEXT` value. Once this environment variable is set, you can start one MPD ring and work with it without affecting other available MPD rings. Set the appropriate `I_MPI_JOB_CONTEXT` value to work with a particular MPD ring. See [Simplified Job Startup Command](#) to learn about an easier way to run several Intel® MPI Library jobs at once.

I\_MPI\_JOB\_TAGGED\_PORT\_OUTPUT

Turn on/off the use of the tagged `mpd` port output.

Syntax

I\_MPI\_JOB\_TAGGED\_PORT\_OUTPUT=<arg>

Arguments

<arg>	Binary indicator
enable   yes   on   1	Turn on the tagged output. This is the default value
disable   no   off   0	Turn off the tagged output

Description

The tagged output format works at the `mpdboot` stage and prevents a failure during startup due to unexpected output from a remote shell like `ssh`. `mpdboot` sets this environment variable to `1` automatically. Set `I_MPI_JOB_TAGGED_PORT_OUTPUT` to `disable` if you do not want to use the new format.

I\_MPI\_MPD\_CHECK\_PYTHON

Turn on/off the Python\* versions check at the MPD ring startup stage.

Syntax

I\_MPI\_MPD\_CHECK\_PYTHON=<arg>

Arguments

<arg>	Binary indicator
-------	------------------

<code>enable   yes   on   1</code>	Check for Python version compatibility
<code>disable   no   off   0</code>	Do not check the Python version compatibility. This is the default value

Description

Set this environment variable to `enable` compatibility checking of Python versions installed on the cluster nodes. This may lead to increased MPD ring startup time. The MPD behavior is undefined if incompatible Python versions are installed on the cluster.

If `I_MPI_MPD_CHECK_PYTHON` is set to `enable` and the compatibility check fails, `mpdboot` exits abnormally and print a diagnostic message. An MPD ring is not started.

I\_MPI\_MPD\_RSH

Set the remote shell to start `mpd` daemons.

Syntax

`I_MPI_MPD_RSH =<arg>`

Arguments

<code>&lt;arg&gt;</code>	String parameter
<code>&lt;remote shell&gt;</code>	The remote shell

Description

Set this environment variable to define the default setting for the `--rsh mpdboot` option. If `--rsh` is explicitly called in the command line, the `I_MPI_MPD_RSH` environment variable has no effect. The `--rsh` option assumes the value of the `I_MPI_MPD_RSH` environment variable if this variable is defined.

I\_MPI\_MPD\_TMPDIR

TMPDIR

Set a temporary directory for the MPD subsystem.

Syntax

`I_MPI_MPD_TMPDIR=<arg>`

`TMPDIR=<arg>`

Arguments

<code>&lt;arg&gt;</code>	String parameter
<code>&lt;directory name&gt;</code>	A string that points to a scratch space location. The default value is <code>/tmp</code>

Description

Set one of these environment variables to specify an alternative scratch space location. The MPD subsystem creates its own files in the directory specified by these environment variables. If both environment variables point to valid directories, the value of the `TMPDIR` environment variable is ignored.

**NOTE:** The `mpd2.console_*` file full path length can be limited in some operating systems. You hit this limitation if you get the following diagnostic message: `socket.error: AF_UNIX path too long`. Decrease the length of the `<directory name>` string to avoid this issue.

**NOTE:** If `<arg>` points to a distributed file system (PANFS, PVFS, etc.), the `mpd` demons may not start. If this happens, set the `I_MPI_MPD_TMPDIR` and `TMPDIR` to point to a standard file system (ext2, ext3, NFS, etc.).

I\_MPI\_MPD\_CLEAN\_LOG

Control the removal of the log file upon MPD termination.

Syntax

`I_MPI_MPD_CLEAN_LOG=<value>`

Arguments

<code>&lt;value&gt;</code>	Define the value
<code>enable   yes   on   1</code>	Remove the log file
<code>disable   no   off   0</code>	Keep the log file. This is the default value

Description

Set this environment variable to define the `mpdallexit` behavior. If you enable this environment variable, the `mpdallexit` removes the log file created during its execution. If you disable this environment variable, the `mpdallexit` keeps the log file.

**NOTE:**

## 2.5 Processor Information Utility

### cpuinfo

The **cpuinfo** utility provides processor architecture information.

#### Syntax

```
cpuinfo [[-]<options>]]
```

#### Arguments

<b>&lt;options&gt;</b>	Sequence of one-letter options. Each option is responsible for the specific part of printed data
<b>g</b>	General information about single cluster node
<b>i</b>	Logical processors identification
<b>d</b>	Node decomposition table
<b>c</b>	Cache sharing by logical processors
<b>s</b>	Microprocessor signature hexadecimal fields (Intel platform notation)
<b>f</b>	Microprocessor feature flags (Intel platform notation)
<b>A</b>	Replacer of all available options union
<b>gidc</b>	Default sequence
<b>?</b>	Utility usage info

#### Description

The **cpuinfo** utility prints out the processor architecture information that can be used to define suitable process pinning settings. The output consists of a number of tables. Each table corresponds to one of the single options listed in the arguments table. See the following examples.

- **General information about single node** shows the processor product name, number of packages/sockets on the node, core and threads numbers on the node and within each package, and SMT mode enabling.

- **Logical processor identification** table identifies threads, cores, and packages of each logical processor accordingly.

*Processor* – logical processor number.

*Thread Id* – unique processor identifier within a core.

*Core Id* – unique core identifier within a package.

*Package Id* – unique package identifier within a node.

- **Node decomposition table** shows the node contents. Each entry contains the information on packages, cores, and logical processors.

*Package Id* – physical package identifier.

*Cores Id* – list of core identifiers that belong to this package.

*Processors Id* – list of processors that belong to this package. This list order directly corresponds to the core list. A group of processors enclosed in brackets belongs to one core.

- **Cache sharing by logical processors** shows information of sizes and processors groups, which share particular cache level.

*Size* – cache size in bytes.

*Processors* – a list of processor groups enclosed in the parentheses those share this cache or *no sharing* otherwise.

- **Microprocessor signature** table shows signature values: extended family, extended model, family, model, type, and stepping.
- **Microprocessor feature flags** indicate what features the microprocessor supports. The Intel platform notation is used.

**NOTE:** The architecture information is available on systems based on the IA-32 and Intel® 64 architectures.

The `cpuinfo` utility is available for both Intel® microprocessors and non-Intel microprocessors, but it may provide only partial information about non-Intel microprocessors.

### Examples

`cpuinfo` output for the processor of Intel® microarchitecture code name Sandy Bridge:

```
$ cpuinfo A
```

```
Intel(R) Processor information utility, Version 4.0 Update 3 Build 20110526
Copyright (C) 2005-2011 Intel Corporation. All rights reserved.
```

```
===== Processor composition =====
Processor name      : Genuine Intel(R)
Packages(sockets)  : 2
Cores               : 16
Processors(CPU)    : 32
Cores per package   : 8
Threads per core    : 2
```

```
===== Processor identification =====
```

Processor	Thread Id.	Core Id.	Package Id.
0	0	0	0
1	0	1	0
2	0	2	0
3	0	3	0
4	0	4	0
5	0	5	0
6	0	6	0
7	0	7	0
8	0	0	1
9	0	1	1
10	0	2	1
11	0	3	1
12	0	4	1
13	0	5	1
14	0	6	1
15	0	7	1
16	1	0	0
17	1	1	0
18	1	2	0
19	1	3	0
20	1	4	0
21	1	5	0
22	1	6	0
23	1	7	0
24	1	0	1
25	1	1	1
26	1	2	1
27	1	3	1
28	1	4	1
29	1	5	1
30	1	6	1
31	1	7	1

## ===== Placement on packages =====

Package Id.	Core Id.	Processors
0	0,1,2,3,4,5,6,7	(0,16)(1,17)(2,18)(3,19)(4,20)(5,21)(6,22)(7,23)
1	0,1,2,3,4,5,6,7	(8,24)(9,25)(10,26)(11,27)(12,28)(13,29)(14,30)(15,31)

## ===== Cache sharing =====

Cache	Size	Processors
L1	32 KB	(0,16)(1,17)(2,18)(3,19)(4,20)(5,21)(6,22)(7,23)(8,24)(9,25)(10,26)(11,27)(12,28)(13,29)(14,30)(15,31)
L2	256 KB	(0,16)(1,17)(2,18)(3,19)(4,20)(5,21)(6,22)(7,23)(8,24)(9,25)(10,26)(11,27)(12,28)(13,29)(14,30)(15,31)
L3	20 MB	(0,1,2,3,4,5,6,7,16,17,18,19,20,21,22,23)(8,9,10,11,12,13,14,15,24,25,26,27,28,29,30,31)

## ===== Processor Signature =====

xFamily	xModel	Type	Family	Model	Stepping
00	0	2	6	d	3

## ===== Processor Feature Flags =====

SSE3	PCLMULDQ	DTES64	MONITOR	DS-CPL	VMX	SMX	EIST	TM2	SSSE3	CNXT-ID	FMA	CX16	xTPR
1	1	1	1	1	1	1	1	1	1	0	0	1	1

PDCM	PCID	DCA	SSE4.1	SSE4.2	x2APIC	MOVBE	POPCNT	TSC-DEADLINE	AES	XSAVE	OSXSAVE	AVX
1	1	1	1	1	1	0	1	1	1	1	1	1

FPU	VME	DE	PSE	TSC	MSR	PAE	MCE	CX8	APIC	SEP	MTRR	PGE	MCA	CMOV	PAT	PSE-36
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

PSN	CLFSH	DS	ACPI	MMX	FXSR	SSE	SSE2	SS	HTT	TM	PBE
0	1	1	1	1	1	1	1	1	1	1	1

## 3 Tuning Reference

The Intel® MPI Library provides an automatic tuning utility to help you select optimal values for many environment variables that can be used to influence program behavior and performance at run time.

### 3.1 Automatic Tuning Utility

#### mpitune

Use the `mpitune` utility to find optimal settings for the Intel® MPI Library relevant to your cluster configuration or your application.

#### Syntax

```
mpitune [ -a "<application command line>" ] [ -of <file-name> ] \
[ -t "<test_cmd_line>" ] [ -cm ] [ -d ] [ -D ] \
[ -dl [d1[,d2...[,dN]]] ] [ -fl [f1[,f2...[,fN]]] ] [ -er ] \
[ -hf <hostsfile> ] [ -h ] [ -hr {min:max/min:/:max} ] \
[ -i <count> ] [ -mr {min:max/min:/:max} ] [ -od <outputdir> ] \
[ -odr <outputdir> ] [ -r <rshcmd> ] [ -pr {min:max/min:/:max} ] \
[ -sf [file-path] ] [ -ss ] [ -s ] [ -td <dir-path> ] \
[ -tl <minutes> ] [ -mh ] [ -os <opt1,...,optN> ] \
[ -oe <opt1,...,optN> ] [ -V ] [ -vi {percent} ; -vix {X factor} ] \
[ -zb ] [ -t ] [ -so ] [ -ar "reg-expr" ] [ -trf <appoutfile> ] \
[ -m {base|optimized} ] [ -avd {min|max} ] [ -pm {mpd|hydra} ] \
[ -co ] [ -sd ] [ -soc ]
```

or

```
mpitune [ --application "<app_cmd_line>" ] [ --output-file <file-name> ] \
[ --test "<test_cmd_line>" ] [ --cluster-mode ] [ --debug ] \
[ --distinct ] [ --device-list [d1[,d2,... [,dN]]] ] \
[ --fabric-list [f1[,f2...[,fN]]] ] [ --existing-ring ] \
[ --host-file <hostsfile> ] [ --help ] \
[ --host-range {min:max/min:/:max} ] [ --iterations <count> ] \
[ --message-range {min:max/min:/:max} ] \
[ --output-directory <outputdir> ] \
[ --output-directory-results <outputdir> ] [ --rsh <rshcmd> ] \
[ --ppn-range {min:max/min:/:max} ; --perhost-range {min:max/min:/:max} ] \
[ --session-file [file-path] ] [ --show-session ] [ --silent ] \
```

```

[ --temp-directory <dir-path> ] [ --time-limit <minutes> ] \
[ --master-host ] [ --options-set <opt1,...,optN> ] \
[ --options-exclude <opt1,...,optN> ] [ --version ] \
[ --valuable-improvement ; --valuable-improvement-x {X factor} ] \
[ --zero-based ] [ --trace ] [ --scheduler-only ] \
[ --application-regexp \"reg-expr\" ] \
[ --test-regexp-file <appoutfile> ] [ --model {base|optimized} ] \
[ --application-value-direction {min|max} ] \
[ --process-manager {mpd|hydra} ] [ -co ] [ -sd ] [ -soc ]

```

## Arguments

<code>-a \"&lt;app_cmd_line&gt;\"   --application \"&lt;app_cmd_line&gt;\"</code>	Switch on the application-specific mode. Quote the full command line as shown including the backslashes
<code>-of &lt;file-name&gt;   --output-file &lt;file-name&gt;</code>	Specify the name of the application configuration file to be generated in the application-specific mode. By default, use the file name <code>\$PWD/app.conf</code>
<code>-t \"&lt;test_cmd_line&gt;\"   --test \"&lt;test_cmd_line&gt;\"</code>	Replace the default Intel® MPI Benchmarks by the indicated benchmarking program in the cluster-specific mode. Quote the full command line as shown including the backslashes
<code>-cm {exclusive full}   -- cluster-mode {exclusive full}</code>	Set the cluster usage mode  <code>full</code> – maximum number of tasks are executed. This is the default mode  <code>exclusive</code> – only one task is executed on the cluster at a time
<code>-d   --debug</code>	Print out the debug information
<code>-D   --distinct</code>	Tune all options separately from each other. This argument is applicable only for the cluster-specific mode
<code>-dl [d1[,d2...[,dN]]]   --device-list [d1[,d2,... [,dN]]]</code>	Select the device(s) you want to tune. Any previously set fabrics are ignored. By default, use all devices listed in the <code>&lt;installdir&gt;/&lt;arch&gt;/etc/devices.xml</code> file
<code>-fl [f1[,f2...[,fN]]]   --fabric-list [f1[,f2...[,fN]]]</code>	Select the fabric(s) you want to tune. Any previously set devices are ignored. By default, use all fabrics listed in the <code>&lt;installdir&gt;/&lt;arch&gt;/etc/fabrics.xml</code> file
<code>-er   --existing-ring</code>	Use an existing MPD ring. By default, a new MPD ring is created. This argument is applicable only if <code>I_MPI_PROCESS_MANAGER</code> is set to <code>mpd</code> .
<code>-hf &lt;hostsfile&gt;   --host-file &lt;hostsfile&gt;</code>	Specify an alternative host file name. By default, use the <code>\$PWD/mpd.hosts</code>
<code>-h   --help</code>	Display the help message
<code>-hr {min:max/min:/:max}   --host-range {min:max/min:/:max}</code>	Set the range of hosts used for testing. The default minimum value is <code>1</code> . The default maximum value is the number of hosts defined by the <code>mpd.hosts</code> or the existing MPD ring. The <code>min:</code> or <code>:max</code> format uses the default values as appropriate
<code>-i &lt;count&gt;   --iterations &lt;count&gt;</code>	Define how many times to run each tuning step. Higher iteration counts increase the tuning time, but may also

	increase the accuracy of the results. The default value is <b>3</b>
<code>-mr {min:max/min:/:max}  </code> <code>--message-range</code> <code>{min:max/min:/:max}</code>	Set the message size range. The default minimum value is <b>0</b> . The default maximum value is <b>4194304</b> ( <b>4mb</b> ). By default, the values are given in bytes. They can also be given in the following format: <b>16kb</b> , <b>8mb</b> or <b>2gb</b> . The <b>min:</b> or <b>:max</b> format uses the default values as appropriate
<code>-od &lt;outputdir&gt;  </code> <code>--output-directory</code> <code>&lt;outputdir&gt;</code>	Specify the directory name for all output files: log-files, session-files, local host-files and report-files. By default, use the current directory. This directory should be accessible from all hosts
<code>-odr &lt;outputdir&gt;  </code> <code>--output-directory-results</code> <code>&lt;outputdir&gt;</code>	Specify the directory name for the resulting configuration files. By default, use the current directory in the application-specific mode and the <code>&lt;installdir&gt;/&lt;arch&gt;/etc</code> in the cluster-specific mode. If <code>&lt;installdir&gt;/&lt;arch&gt;/etc</code> is unavailable, <code>\$PWD</code> is used as the default value in the cluster-specific mode
<code>-r &lt;rshcmd&gt;   --rsh &lt;rshcmd&gt;</code>	Specify the remote shell used to start daemons (as applicable) and jobs. The default value is <b>ssh</b> .
<code>-pr {min:max/min:/:max}  </code> <code>--ppn-range</code> <code>{min:max/min:/:max}  </code> <code>--perhost-range</code> <code>{min:max/min:/:max}</code>	Set the maximum number of processes per host. The default minimum value is <b>1</b> . The default maximum value is the number of cores of the processor. The <b>min:</b> or <b>:max</b> format uses the default values as appropriate
<code>-sf [file-path]  </code> <code>--session-file [file-path]</code>	Continue the tuning process starting from the state saved in the <code>file-path</code> session file
<code>-ss  </code> <code>--show-session</code>	Show information about the session file and exit. This option works only jointly with the <code>-sf</code> option
<code>-s   --silent</code>	Suppress all diagnostics
<code>-td &lt;dir-path&gt;  </code> <code>--temp-directory &lt;dir-path&gt;</code>	Specify a directory name for the temporary data. Use <code>\$PWD/mpiunertemp</code> by default. This directory should be accessible from all hosts
<code>-tl &lt;minutes&gt;  </code> <code>--time-limit &lt;minutes&gt;</code>	Set <code>mpitune</code> execution time limit in minutes. The default value is <b>0</b> , which means no limitations
<code>-mh /</code> <code>--master-host</code>	Dedicate a single host to run the <code>mpitune</code>
<code>-os &lt;opt1,...,optN&gt; </code> <code>--options-set &lt;opt1,...,optN&gt;</code>	Use <code>mpitune</code> to tune the only required options you have set in the option values
<code>-oe &lt;opt1,...,optN&gt; </code> <code>--options-exclude</code> <code>&lt;opt1,...,optN&gt;</code>	Exclude the settings of the indicated Intel® MPI Library options from the tuning process
<code>-V   --version</code>	Print out the version information
<code>-vi {percent}  </code> <code>--valuable-improvement</code> <code>{percent}</code> <code>-vix {X factor}  </code> <code>--valuable-improvement-x</code>	Control the threshold for performance improvement. The default threshold is <b>3%</b>

<code>{X factor}</code>	
<code>-zb   --zero-based</code>	Set zero as the base for all options before tuning. This argument is applicable only for the cluster-specific mode
<code>-t   --trace</code>	Print out error information such as error codes and tuner traceback
<code>-so   --scheduler-only</code>	Create the list of tasks to be executed, display the tasks, and terminate execution
<code>-ar "reg-expr"   --application-regexp "reg-expr"</code>	Use <code>reg-expr</code> to determine the performance expectations of the application. This option is applicable only for the application-specific mode. The <code>reg-expr</code> setting should contain only one group of numeric values which is used by <code>mpitune</code> for analysis. Use backslash for symbols when setting the value of this argument in accordance with the operating system requirements
<code>-trf &lt;appoutfile&gt;   --test-regexp-file &lt;appoutfile&gt;</code>	Use a test output file to check the correctness of the regular expression. This argument is applicable only for the cluster-specific mode when you use the <code>-ar</code> option
<code>-m {base optimized}   --model {base optimized}</code>	Specify the search model: <ul style="list-style-type: none"> <li>Set <code>base</code> to use the old model</li> <li>Set <code>optimized</code> to use the new faster search model. This is the default value</li> </ul>
<code>-avd {min max}   --application-value- direction {min max}</code>	Specify the direction of the value optimization : <ul style="list-style-type: none"> <li>Set <code>min</code> to specify that lower is better. For example, use this value when optimizing the wall time</li> <li>Set <code>max</code> to specify that higher is better. For example, use this value when optimizing the solver ratio</li> </ul>
<code>-pm {mpd hydra}   --process-manager {mpd hydra}</code>	Specify the process manager used to run the benchmarks. The default value is <code>hydra</code>
<code>-co   --collectives-only</code>	Tune collective operations only
<code>-sd   --save-defaults</code>	Use <code>mpitune</code> to save the default values of the Intel® MPI Library options
<code>-soc   --skip-options-check</code>	Specify whether to check the command line options

### Deprecated Options

Deprecated Option	New Option
<code>--outdir</code>	<code>-od   --output-directory</code>
<code>--verbose</code>	<code>-d   --debug</code>
<code>--file</code>	<code>-hf   --host-file</code>
<code>--logs</code>	<code>-lf   --log-file</code>
<code>--app</code>	<code>-a   --application</code>

### Description

Use the `mpitune` utility to create a set of Intel® MPI Library configuration files that contain optimal settings for a particular cluster or application. You can reuse these configuration files in the `mpirun` job launcher by using the `-tune` option. If configuration files from previous `mpitune` sessions exist, `mpitune` creates a copy of the existing files before starting execution.

The MPI tuner utility operates in two modes:

- Cluster-specific, evaluating a given cluster environment using either the Intel® MPI Benchmarks or a user-provided benchmarking program to find the most suitable configuration of the Intel® MPI Library. This mode is used by default.
- Application-specific, evaluating the performance of a given MPI application to find the best configuration for the Intel® MPI Library for the particular application. Application tuning is enabled by the `--application` command line option.

## 3.1.1 Cluster-specific Tuning

Run this utility once after the Intel® MPI Library installation and after every cluster configuration change (processor or memory upgrade, network reconfiguration, etc.). Do this under the user account that was used for the Intel® MPI Library installation or appropriately set the tuner data directory through the `--output-directory` option and the results directory through the `--output-directory-results` option.

If there are any configuration files in the `<installdir>/<arch>/etc` directory, the recorded Intel® MPI Library configuration settings are used automatically by `mpiexec` with the `-tune` option.

For example:

- Collect configuration settings for the cluster hosts listed in the `./mpd.hosts` file by using the Intel® MPI Benchmarks

```
$ mpitune
```

- Use the optimal recorded values when running on the cluster

```
$ mpiexec -tune -n 32 ./myprog
```

The job launcher finds a proper set of configuration options based on the following execution conditions: communication fabrics, number of hosts and processes, etc. If you have write access permission for `<installdir>/<arch>/etc`, all generated files are saved in this directory; otherwise the current working directory is used.

### 3.1.1.1 Replacing the Default Benchmark

This tuning feature is an extension of the cluster-specific tuning mode in which you specify a benchmarking application that will be used for tuning.

The Intel® MPI Benchmarks executables, which are more optimized for Intel microprocessors than for non-Intel microprocessors, are used by default. This may result in different tuning settings on Intel microprocessors than on non-Intel microprocessors.

For example:

1. Collect the configuration settings for the cluster hosts listed in the `./mpd.hosts` file by using the desired benchmarking program

```
$ mpitune --test \"benchmark -param1 -param2\"
```

2. Use the optimal recorded values for your cluster

```
$ mpiexec -tune -n 32 ./myprog
```

## 3.1.2 Application-specific Tuning

Run the tuning process for any kind of MPI application by specifying its command line to the tuner. Performance is measured as inversed execution time of the given application. To reduce the overall tuning time, use the shortest representative application workload if applicable.

For example:

1. Collect configuration settings for the given application

```
$ mpitune --application \"mpiexec -n 32 ./myprog\" -of ./myprog.conf
```

2. Use the optimal recorded values for your application

```
$ mpiexec -tune ./myprog.conf -n 32 ./myprog
```

Based on the default tuning rules, the automated tuning utility evaluates a full set of the library configuration parameters to minimize the application execution time. By default, all generated files will be saved in the current working directory.

**NOTE:** The resulting application configuration file contains the optimal Intel® MPI Library parameters for this application only. If you want to tune the Intel® MPI Library for the same application in a different configuration (number of hosts, workload, etc.), you may need to rerun the automated tuning utility by using the desired configuration.

The automated tuning utility will overwrite the existing application configuration files by default. You should use a naming convention for your various application files to create and select the correct file when you need it.

## 3.1.3 Tuning Utility Output

Upon completion of the tuning process, the Intel® MPI Library tuning utility records the chosen values in the configuration file in the following format:

```
-genv I_MPI_DYNAMIC_CONNECTION 1
-genv I_MPI_ADJUST_REDUCE 1:0-8
```

The Intel MPI Library tuning utility ignores any environment variables that have no effect on the application when the difference between probes is at the noise level (1%). In this case, the utility does not set the environment variable and preserves the default library heuristics.

In the case of the tuning application having significant run-to-run performance variation, the Intel MPI Library tuning utility might select divergent values for the same environment variable under the same conditions. To improve decision accuracy, increase the number of iterations for each test run with the `--iterations` command line option. The default value for the iteration number is 3.

## 3.2 Process Pinning

Use this feature to pin particular MPI process to a corresponding CPU and avoid undesired process migration. This feature is available on operating systems that provide the necessary kernel interfaces.

### 3.2.1 Process Identification

Two schemes are used to identify logical processors in a system:

1. System-defined logical enumeration

- 2. Topological enumeration based on three-level hierarchical identification through triplets (package/socket, core, thread)

The number of a logical CPU is defined as the corresponding position of this CPU bit in the kernel affinity bit-mask. Use the `cpuinfo` utility, provided with your Intel MPI Library installation, or the `cat /proc/cpuinfo` command to find out the logical CPU numbers.

The three-level hierarchical identification uses triplets that provide information about processor location and their order. The triplets are hierarchically ordered (package, core, and thread).

See the example below for one possible processor numbering scenario with two sockets, four cores (two cores per socket), and eight logical processors (two processors per core).

**NOTE:** Logical and topological enumerations are not the same.

Table 3.2-1 Logical Enumeration

0	4	1	5	2	6	3	7
---	---	---	---	---	---	---	---

Table 3.2-2 Hierarchical Levels

Socket	0	0	0	0	1	1	1	1
Core	0	0	1	1	0	0	1	1
Thread	0	1	0	1	0	1	0	1

Table 3.2-3 Topological Enumeration

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

Use the `cpuinfo` utility to identify the correspondence between the logical and topological enumerations. See [Processor Information Utility](#) for more details.

## 3.2.2 Environment Variables

### `I_MPI_PIN`

Turn on/off process pinning.

#### Syntax

`I_MPI_PIN=<arg>`

#### Arguments

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Enable process pinning. This is the default value
<code>disable   no   off   0</code>	Disable processes pinning

#### Description

Set this environment variable to turn off the process pinning feature of the Intel® MPI Library.

### `I_MPI_PIN_MODE`

Choose the pinning method.

## Syntax

`I_MPI_PIN_MODE=<pinmode>`

## Arguments

<code>&lt;pinmode&gt;</code>	Choose the CPU pinning mode
<code>mpd</code>	Pin processes inside the MPD. This is the default value on the SGI* Altix* platform
<code>lib</code>	Pin processes inside the Intel MPI Library. This is the default value on other platforms

## Description

Set the `I_MPI_PIN_MODE` environment variable to choose the pinning method. This environment variable is valid only if the `I_MPI_PIN` environment variable is enabled.

Set this environment variable to `lib` to make the Intel® MPI Library pin the processes. In this mode there is no chance to co-locate the process CPU and its memory.

Set the `I_MPI_PIN_MODE` environment variable to `mpd` to make the `mpd` daemon pin processes through system specific means, if they are available. The pinning is done before the MPI process launch. Therefore, it is possible to co-locate the process CPU and memory in this case. This pinning method has an advantage over a system with Non-Uniform Memory Architecture (NUMA) like SGI\* Altix\*. Under NUMA, a processor can access its own local memory faster than non-local memory.

**NOTE:** It is not recommended to change the default settings.

## `I_MPI_PIN_PROCESSOR_LIST` (`I_MPI_PIN_PROCS`)

Define a processor subset and the mapping rules for MPI processes within this subset.

## Syntax

`I_MPI_PIN_PROCESSOR_LIST=<value>`

The environment variable value has three syntax forms:

1. `<proclist>`
2. `[ <procset> ] [ :[ grain=<grain> ] [ ,shift=<shift> ] \`  
`[ ,preoffset=<preoffset> ] [ ,postoffset=<postoffset> ]`
3. `[ <procset> ][ :map=<map> ]`

## Deprecated Syntax

`I_MPI_PIN_PROCS=<proclist>`

**NOTE:** The `postoffset` keyword has `offset` alias.

**NOTE:** The second form of the pinning procedure has three steps:

1. Cyclic shift of the source processor list on `preoffset*grain` value.
2. Round robin shift of the list derived on the first step on `shift*grain` value.
3. Cyclic shift of the list derived on the second step on the `postoffset*grain` value.

The resulting processor list is used for the consecutive mapping of MPI processes (i-th rank is mapped to the i-th list member).

**NOTE:** The `grain`, `shift`, `preoffset`, and `postoffset` parameters have a unified definition style.

This environment variable is available for both Intel® and non-Intel microprocessors, but it may perform additional optimizations for Intel microprocessors than it performs for non-Intel microprocessors.

### Arguments

<code>&lt;proclist&gt;</code>	A comma-separated list of logical processor numbers and/or ranges of processors. The process with the i-th rank is pinned to the i-th processor in the list. The number should not exceed the amount of processors on a node
<code>&lt;l&gt;</code>	Processor with logical number <code>&lt;l&gt;</code>
<code>&lt;l&gt;-&lt;m&gt;</code>	Range of processors with logical numbers from <code>&lt;l&gt;</code> to <code>&lt;m&gt;</code>
<code>&lt;k&gt;, &lt;l&gt;-&lt;m&gt;</code>	Processors <code>&lt;k&gt;</code> , as well as <code>&lt;l&gt;</code> through <code>&lt;m&gt;</code>

<code>&lt;procset&gt;</code>	Specify a processor subset based on the topological numeration. The default value is <code>allcores</code>
<code>all</code>	All logical processors. This subset is defined to be the number of CPUs on a node
<code>allcores</code>	All cores (physical CPUs). This subset is defined to be the number of cores on a node. This is the default value. If Intel® Hyper-Threading Technology is disabled, <code>allcores</code> equals to <code>all</code>
<code>allsockets</code>	All packages/sockets. This subset is defined to be the number of sockets on a node

<code>&lt;map&gt;</code>	The mapping pattern used for process placement
<code>bunch</code>	The processes are mapped as close as possible on the sockets
<code>scatter</code>	The processes are mapped as remotely as possible so as not to share common resources: FSB, caches, core
<code>spread</code>	The processes are mapped consecutively with the possibility not to share common resources

<code>&lt;grain&gt;</code>	Specify the pinning granularity cell for a defined <code>&lt;procset&gt;</code> . The minimal <code>&lt;grain&gt;</code> is a single element of the <code>&lt;procset&gt;</code> . The maximal grain is the number of <code>&lt;procset&gt;</code> elements in a socket. The <code>&lt;grain&gt;</code> value must be a multiple of the <code>&lt;procset&gt;</code> value. Otherwise, minimal grain is assumed. The default value is the minimal <code>&lt;grain&gt;</code>
<code>&lt;shift&gt;</code>	Specify the round robin shift of the granularity cells for the <code>&lt;procset&gt;</code> . <code>&lt;shift&gt;</code> is measured in the defined <code>&lt;grain&gt;</code> units. The <code>&lt;shift&gt;</code> value must be positive integer. Otherwise, no shift is performed. The default value is no shift
<code>&lt;preoffset&gt;</code>	Specify the cyclic shift of the processor subset <code>&lt;procset&gt;</code> defined before the round robin shifting on the <code>&lt;preoffset&gt;</code> value. The value is measured in the defined <code>&lt;grain&gt;</code> units. The <code>&lt;preoffset&gt;</code> value must be non negative integer. Otherwise, no shift is performed. The default

	value is no shift
<code>&lt;postoffset&gt;</code>	Specify the cyclic shift of the processor subset <code>&lt;procset&gt;</code> derived after round robin shifting on the <code>&lt;postoffset&gt;</code> value. The value is measured in the defined <code>&lt;grain&gt;</code> units. The <code>&lt;postoffset&gt;</code> value must be non-negative integer. Otherwise no shift is performed. The default value is no shift

<code>&lt;n&gt;</code>	Specify an explicit value of the corresponding parameters previously mentioned. <code>&lt;n&gt;</code> is non-negative integer
<code>fine</code>	Specify the minimal value of the corresponding parameter
<code>core</code>	Specify the parameter value equal to the amount of the corresponding parameter units contained in one core
<code>cache1</code>	Specify the parameter value equal to the amount of the corresponding parameter units that share an L1 cache
<code>cache2</code>	Specify the parameter value equal to the amount of the corresponding parameter units that share an L2 cache
<code>cache3</code>	Specify the parameter value equal to the amount of the corresponding parameter units that share an L3 cache
<code>cache</code>	The largest value among <code>cache1</code> , <code>cache2</code> , and <code>cache3</code>
<code>socket</code>   <code>sock</code>	Specify the parameter value equal to the amount of the corresponding parameter units contained in one physical package/socket
<code>half</code>   <code>mid</code>	Specify the parameter value equal to <code>socket/2</code>
<code>third</code>	Specify the parameter value equal to <code>socket/3</code>
<code>quarter</code>	Specify the parameter value equal to <code>socket/4</code>
<code>octavo</code>	Specify the parameter value equal to <code>socket/8</code>

## Description

Set the `I_MPI_PIN_PROCESSOR_LIST` environment variable to define the processor placement. To avoid conflicts with differing shell versions, the environment variable value may need to be enclosed in quotes.

**NOTE:** This environment variable is valid only if `I_MPI_PIN` is enabled.

The `I_MPI_PIN_PROCESSOR_LIST` environment variable has three different syntax variants:

1. Explicit processor list. This comma-separated list is defined in terms of logical processor numbers. The relative node rank of a process is an index to the processor list such that the *i*-th process is pinned on *i*-th list member. This permits the definition of any process placement on the CPUs.

For example, process mapping for `I_MPI_PROCESSOR_LIST=p0,p1,p2,...,pn` is as follows:

Rank on a node	0	1	2	...	n-1	N
Logical CPU	p0	p1	p2	...	pn-1	Pn

2. `grain/shift/offset` mapping. This method provides cyclic shift of a defined `grain` along the processor list with steps equal to `shift*grain` and a single shift on `offset*grain` at the end. This shifting action is repeated `shift` times.

For example: grain = 2 logical processors, shift = 3 grains, offset = 0.

Legend:

gray – MPI process grains

A) red – processor grains chosen on the 1<sup>st</sup> pass

B) cyan – processor grains chosen on the 2<sup>nd</sup> pass

C) green – processor grains chosen on the final 3<sup>rd</sup> pass

D) Final map table ordered by MPI ranks

A)

0 1			2 3			...	2n-2 2n-1		
0 1	2 3	4 5	6 7	8 9	10 11	...	6n-6 6n-5	6n-4 6n-3	6n-2 6n-1

B)

0 1	2n 2n+1		2 3	2n+2 2n+3		...	2n-2 2n-1	4n-2 4n-1	
0 1	2 3	4 5	6 7	8 9	10 11	...	6n-6 6n-5	6n-4 6n-3	6n-2 6n-1

C)

0 1	2n 2n+1	4n 4n+1	2 3	2n+2 2n+3	4n+2 4n+3	...	2n-2 2n-1	4n-2 4n-1	6n-2 6n-1
0 1	2 3	4 5	6 7	8 9	10 11	...	6n-6 6n-5	6n-4 6n-3	6n-2 6n-1

D)

0 1	2 3	...	2n-2 2n-1	2n 2n+1	2n+2 2n+3	...	4n-2 4n-1	4n 4n+1	4n+2 4n+3	...	6n-2 6n-1
0 1	6 7	...	6n-6 6n-5	2 3	8 9	...	6n-4 6n-3	4 5	10 11	...	6n-2 6n-1

3. Predefined mapping scenario. In this case popular process pinning schemes are defined as keywords selectable at runtime. There are two such scenarios: **bunch** and **scatter**.

In the **bunch** scenario the processes are mapped proportionally to sockets as closely as possible. This makes sense for partial processor loading. In this case the number of processes is less than the number of processors.

In the **scatter** scenario the processes are mapped as remotely as possible so as not to share common resources: FSB, caches, cores.

In the example below there are two sockets, four cores per socket, one logical CPU per core, and two cores per shared cache.

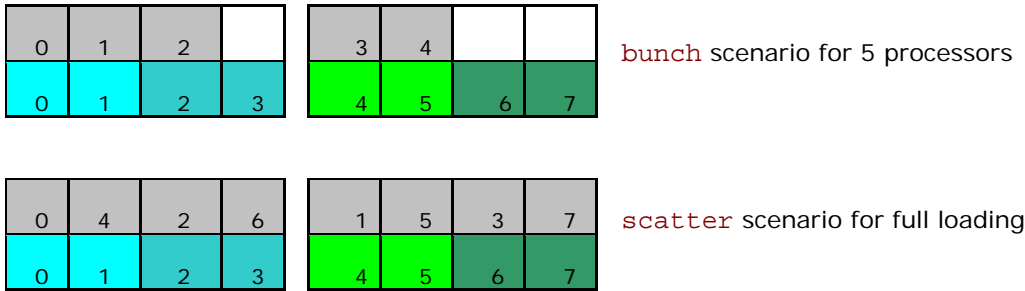
Legend:

gray – MPI processes

cyan – 1<sup>st</sup> socket processors

green – 2<sup>nd</sup> socket processors

Same color defines a processor pair sharing a cache



Examples

- 1. To pin the processes to CPU0 and CPU3 on each node globally, use the following command:  
\$ mpirun -genv I\_MPI\_PIN\_PROCESSOR\_LIST 0,3 \  
-n <# of processes> <executable>
- 2. To pin the processes to different CPUs on each node individually (CPU0 and CPU3 on host1 and CPU0, CPU1 and CPU3 on host2), use the following command:  
\$ mpirun -host host1 -env I\_MPI\_PIN\_PROCESSOR\_LIST 0,3 \  
-n <# of processes> <executable> : \  
-host host2 -env I\_MPI\_PIN\_PROCESSOR\_LIST 1,2,3 \  
-n <# of processes> <executable>
- 3. To print extra debug information about process pinning, use the following command:  
\$ mpirun -genv I\_MPI\_DEBUG 4 -m -host host1 \  
-env I\_MPI\_PIN\_PROCESSOR\_LIST 0,3 -n <# of processes> <executable> :\  
-host host2 -env I\_MPI\_PIN\_PROCESSOR\_LIST 1,2,3 \  
-n <# of processes> <executable>

**NOTE:** If the number of processes is greater than the number of CPUs used for pinning, the process list is wrapped around to the start of the processor list.

I\_MPI\_PIN\_CELL

Set this environment variable to define the pinning resolution granularity. I\_MPI\_PIN\_CELL specifies the minimal processor cell allocated when an MPI process is running.

Syntax

I\_MPI\_PIN\_CELL=<cell>

Arguments

<cell>	Specify the resolution granularity
unit	Basic processor unit (logical CPU)
core	Physical processor core

Description

Set this environment variable to define the processor subset used when a process is running. You can choose from two scenarios:

- all possible CPUs in a system (unit value)
- all cores in a system (core value)

The environment variable has effect on both pinning kinds:

- one-to-one pinning through the `I_MPI_PIN_PROCESSOR_LIST` environment variable
- one-to-many pinning through the `I_MPI_PIN_DOMAIN` environment variable

The default value rules are:

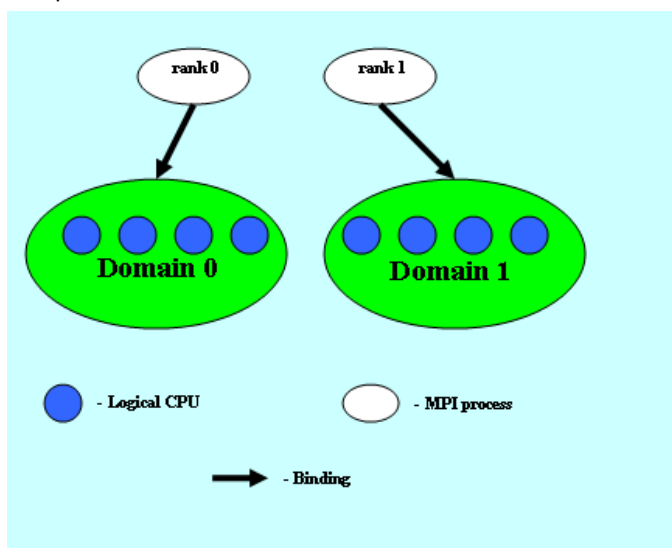
- If you use `I_MPI_PIN_DOMAIN`, then the cell granularity is `unit`.
- If you use `I_MPI_PIN_PROCESSOR_LIST`, then the following rules apply:
  - When the number of processes is greater than the number of cores, the cell granularity is `unit`.
  - When the number of processes is equal to or less than the number of cores, the cell granularity is `core`.

**NOTE:** The `core` value is not affected by the enabling/disabling of Hyper-threading technology in a system.

## 3.2.3 Interoperability with OpenMP\*

### `I_MPI_PIN_DOMAIN`

The Intel® MPI Library provides an additional environment variable to control process pinning for hybrid Intel MPI Library applications. This environment variable is used to define a number of non-overlapping subsets (domains) of logical processors on a node, and a set of rules on how MPI processes are bound to these domains by the following formula: *one MPI process per one domain*. See the picture below.



**Picture 3.2-1 Domain Example**

Each MPI process can create a number of children threads for running within the corresponding domain. The process threads can freely migrate from one logical processor to another within the particular domain. There are no domains defined by default so they should all be created explicitly.

If the `I_MPI_PIN_DOMAIN` environment variable is defined, then the `I_MPI_PIN_PROCESSOR_LIST` environment variable setting is ignored.

If the `I_MPI_PIN_DOMAIN` environment variable is not defined, then MPI processes are pinned according to the current value of the `I_MPI_PIN_PROCESSOR_LIST` environment variable.

The `I_MPI_PIN_DOMAIN` environment variable has the following syntax forms:

1. Domain description through multi-core terms
2. Domain description through domain size and domain member layout
3. Explicit domain description through bit mask

The following tables describe these syntax forms.

## Multi-core Shape

`I_MPI_PIN_DOMAIN=<mc-shape>`

<code>&lt;mc-shape&gt;</code>	Define domains through multi-core terms
<code>core</code>	Each domain consists of the logical processors that share a particular core. The number of domains on a node is equal to the number of cores on the node
<code>socket   sock</code>	Each domain consists of the logical processors that share a particular socket. The number of domains on a node is equal to the number of sockets on the node. This is the recommended value.
<code>node</code>	All logical processors on a node are arranged into a single domain
<code>cache1</code>	Logical processors that share a particular level 1 cache are arranged into a single domain
<code>cache2</code>	Logical processors that share a particular level 2 cache are arranged into a single domain
<code>cache3</code>	Logical processors that share a particular level 3 cache are arranged into a single domain
<code>cache</code>	The largest domain among <code>cache1</code> , <code>cache2</code> , and <code>cache3</code> is selected

## Explicit Shape

`I_MPI_PIN_DOMAIN=<size>[:<layout>]`

<code>&lt;size&gt;</code>	Define a number of logical processors in each domain (domain size)
<code>omp</code>	The domain size is equal to the <code>OMP_NUM_THREADS</code> environment variable value. If the <code>OMP_NUM_THREADS</code> environment variable is not set, each node is treated as a separate domain.
<code>auto</code>	The domain size is defined by the formula <code>size=#cpu/#proc</code> , where <code>#cpu</code> is the number of logical processors on a node, and <code>#proc</code> is the number of the MPI processes started on a node
<code>&lt;n&gt;</code>	The domain size is defined by a positive decimal number <code>&lt;n&gt;</code>

<code>&lt;layout&gt;</code>	Ordering of domain members. The default value is <code>compact</code>
<code>platform</code>	Domain members are ordered according to their BIOS numbering (platform-depended numbering)
<code>compact</code>	Domain members are located as close to each other as possible in terms of common resources (cores, caches, sockets, etc.). This is the default value
<code>scatter</code>	Domain members are located as far away from each other as possible in terms of common resources (cores, caches, sockets, etc.)

Explicit Domain Mask

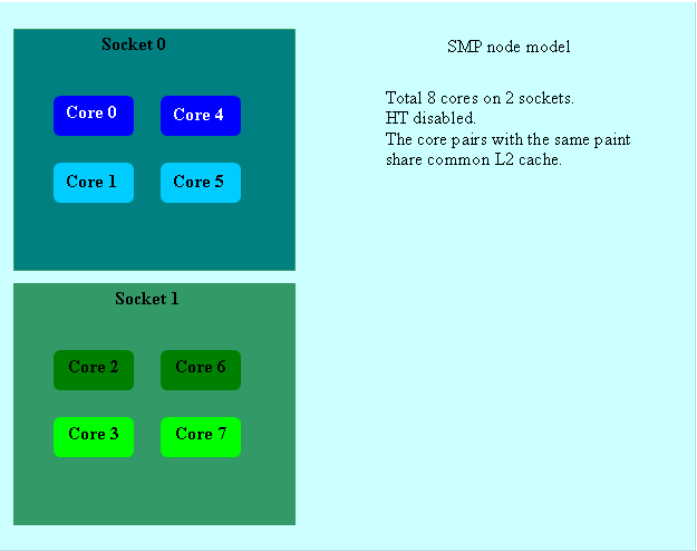
I\_MPI\_PIN\_DOMAIN=<masklist>

<masklist>	Define domains through the comma separated list of hexadecimal numbers (domain masks)
[ m <sub>1</sub> , ... , m <sub>n</sub> ]	Each m <sub>i</sub> number defines one separate domain. The following rule is used: the i <sup>th</sup> logical processor is included into the domain if the corresponding m <sub>i</sub> value is set to 1. All remaining processors are put into a separate domain. BIOS numbering is used

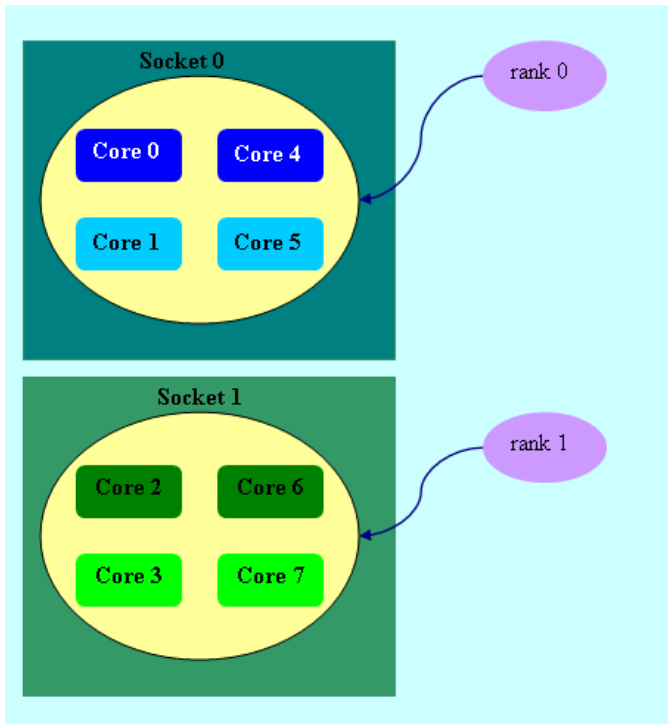
**NOTE:** These options are available for both Intel® and non-Intel microprocessors, but they may perform additional optimizations for Intel microprocessors than they perform for non-Intel microprocessors.

**NOTE:** To pin OpenMP processes/threads inside the domain, the corresponding OpenMP feature (KMP\_AFFINITY environment variable) should be used.

See the following model of an SMP node in the examples below:

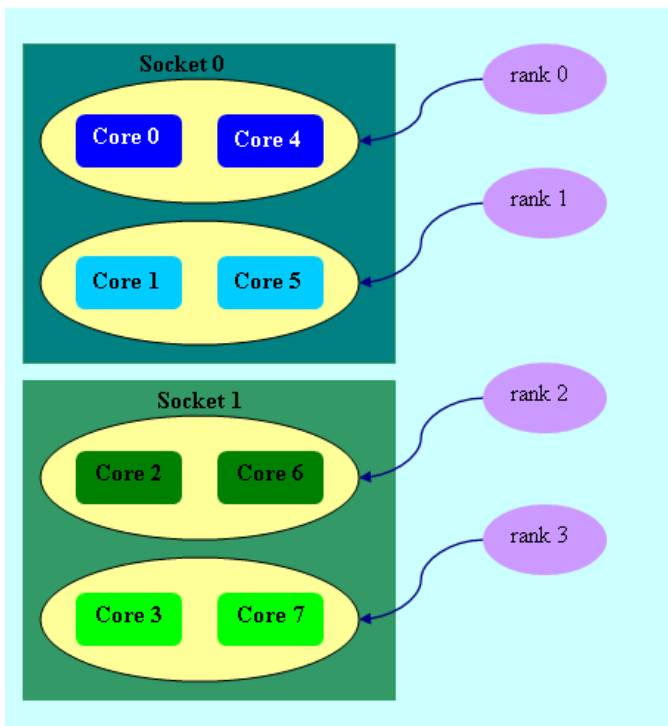


Picture 3.2-2 Model of a Node



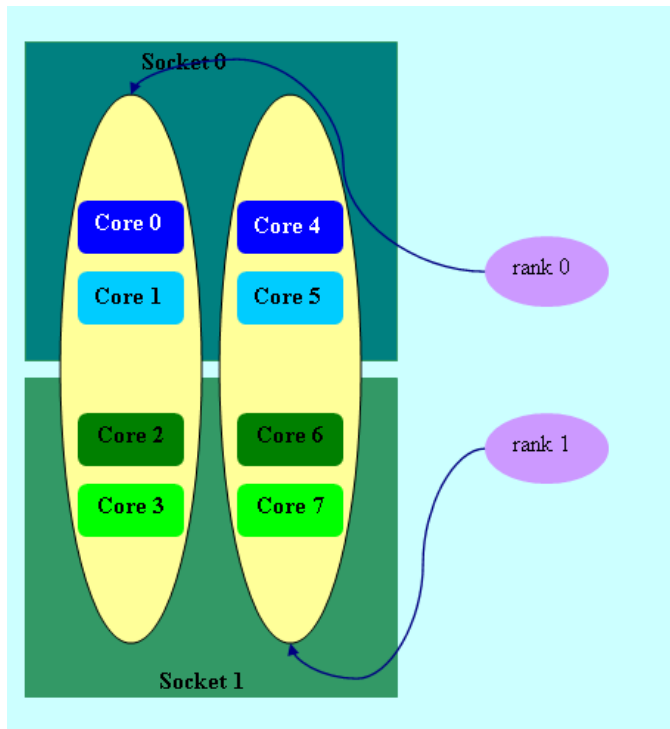
**Picture 3.2-3** `mpiexec -n 2 -env I_MPI_PIN_DOMAIN socket ./a.out`

Two domains are defined according to the number of sockets. Process rank 0 can migrate on all cores on the 0-th socket. Process rank 1 can migrate on all cores on the first socket.



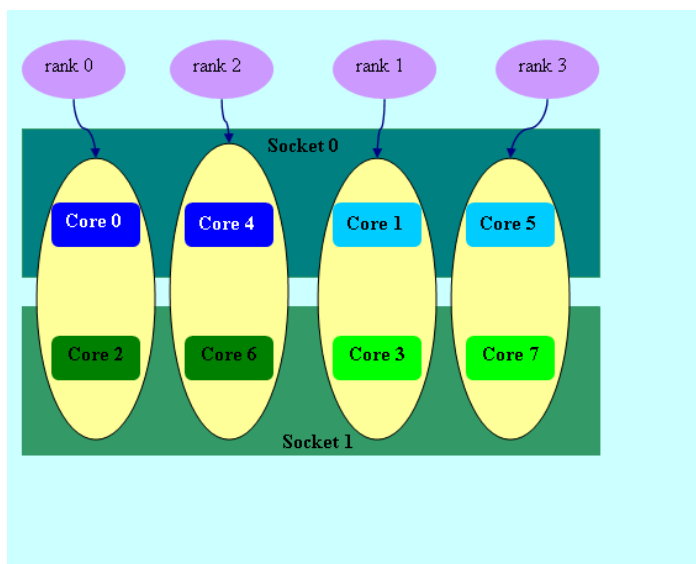
**Picture 3.2-4** `mpiexec -n 4 -env I_MPI_PIN_DOMAIN cache2 ./a.out`

Four domains are defined according to the amount of common L2 caches. Process rank 0 runs on cores {0,4} that share an L2 cache. Process rank 1 runs on cores {1,5} that share an L2 cache as well, and so on.



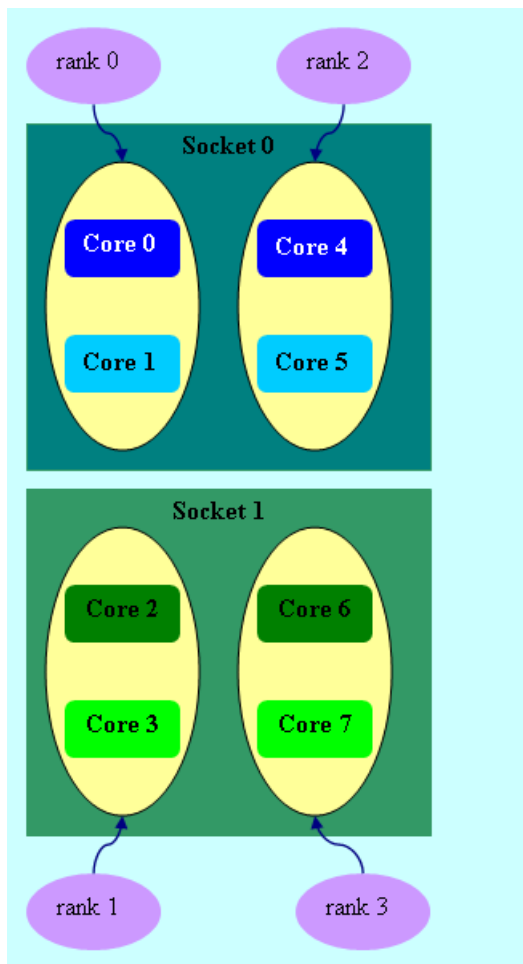
**Picture 3.2-5** `mpiexec -n 2 -env I_MPI_PIN_DOMAIN 4:platform ./a.out`

Two domains with size=4 are defined. The first domain contains cores {0,1,2,3}, and the second domain contains cores {4,5,6,7}. Domain members (cores) have consecutive numbering as defined by the `platform` option.



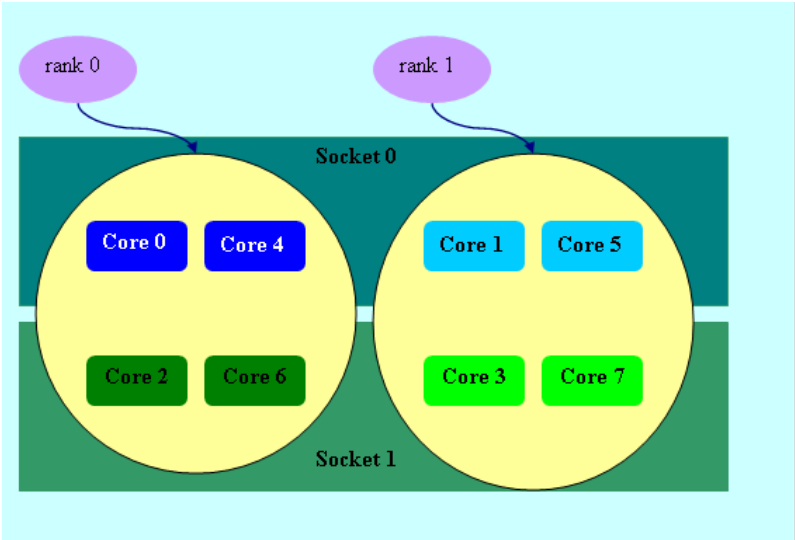
**Picture 3.2-6** `mpiexec -n 4 -env I_MPI_PIN_DOMAIN auto:scatter ./a.out`

Domain size=2 (defined by the number of CPUs=8 / number of processes=4), `scatter` layout. Four domains {0,2}, {1,3}, {4,6}, {5,7} are defined. Domain members do not share any common resources.



**Picture 3.2-7** `mpiexec -n 4 -env I_MPI_PIN_DOMAIN omp:platform ./a.out`  
`setenv OMP_NUM_THREADS=2`

Domain size=2 (defined by `OMP_NUM_THREADS=2`), `platform` layout. Four domains {0,1}, {2,3}, {4,5}, {6,7} are defined. Domain members (cores) have consecutive numbering.



**Picture 3.2-8** `mpiexec -n 2 -env I_MPI_PIN_DOMAIN [55,aa] ./a.out`

The first domain is defined by the 0x55 mask. It contains all cores with even numbers {0,2,4,6}. The second domain is defined by the 0xAA mask. It contains all cores with odd numbers {1,3,5,7}.

### I\_MPI\_PIN\_ORDER

Set this environment variable to define the mapping order for MPI processes to domains as specified by the `I_MPI_PIN_DOMAIN` environment variable.

#### Syntax

`I_MPI_PIN_ORDER=<order>`

#### Arguments

<code>&lt;order&gt;</code>	Specify the ranking order
<code>range</code>	The domains are ordered according to the processor's BIOS numbering. This is a platform-dependent numbering
<code>scatter</code>	The domains are ordered so that adjacent domains have minimal sharing of common resources
<code>compact</code>	The domains are ordered so that adjacent domains share common resources as much as possible. This is the default value

#### Description

The optimal setting for this environment variable is application-specific. If adjacent MPI processes prefer to share common resources, such as cores, caches, sockets, FSB, use the `compact` value. Otherwise, use the `scatter` value. Use the `range` value as needed.

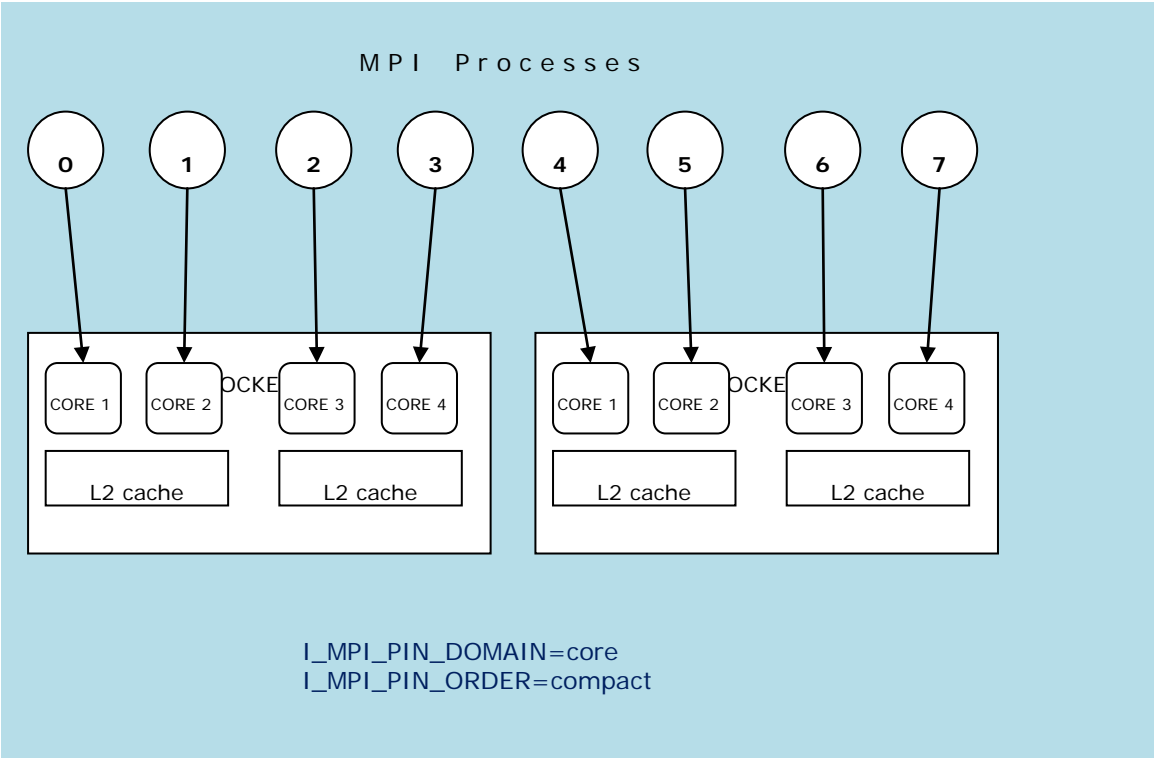
The options `scatter` and `compact` are available for both Intel® and non-Intel microprocessors, but they may perform additional optimizations for Intel microprocessors than they perform for non-Intel microprocessors.

#### Example

Assume we have:

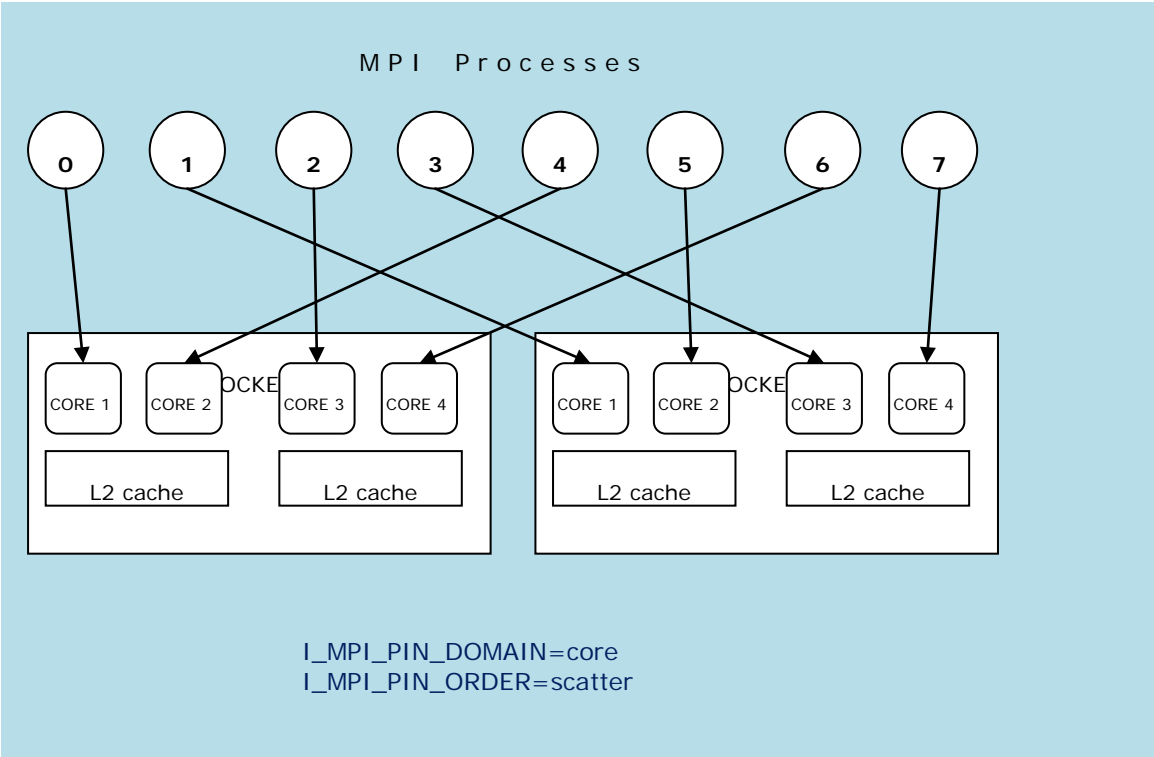
- Two socket node with four cores and a shared L2 cache for corresponding core pairs.
- 8 MPI processes we want to run on the node using
  - o The following settings:  
`I_MPI_PIN_DOMAIN=core`

I\_MPI\_PIN\_ORDER=compact



Picture 3.2-9 Compact Order Example

- o The following settings:  
I\_MPI\_PIN\_DOMAIN=core  
I\_MPI\_PIN\_ORDER=scatter



Picture 3.2-10 Scatter Order Example

## 3.3 Fabrics Control

### 3.3.1 Communication Fabrics Control

#### I\_MPI\_FABRICS

##### (I\_MPI\_DEVICE)

Select the particular network fabrics to be used.

##### Syntax

`I_MPI_FABRICS=<fabric>/<intra-node fabric>:<inter-nodes fabric>`

Where `<fabric> := {shm, dapl, tcp, tmi, ofa}`

`<intra-node fabric> := {shm, dapl, tcp, tmi, ofa}`

`<inter-nodes fabric> := {dapl, tcp, tmi, ofa}`

##### Deprecated Syntax

`I_MPI_DEVICE=<device>[:<provider>]`

##### Arguments

<code>&lt;fabric&gt;</code>	Define a network fabric
<code>shm</code>	Shared-memory
<code>dapl</code>	DAPL-capable network fabrics, such as InfiniBand*, iWarp*, Dolphin*, and XPMEM* (through DAPL*)
<code>tcp</code>	TCP/IP-capable network fabrics, such as Ethernet and InfiniBand* (through IPoIB*)
<code>tmi</code>	TMI-capable network fabrics including Qlogic*, Myrinet*, (through Tag Matching Interface)
<code>ofa</code>	OFA-capable network fabric including InfiniBand* (through OFED* verbs)

##### Correspondence with I\_MPI\_DEVICE

<code>&lt;device&gt;</code>	<code>&lt;fabric&gt;</code>
<code>sock</code>	<code>tcp</code>
<code>shm</code>	<code>shm</code>
<code>ssm</code>	<code>shm:tcp</code>
<code>rdma</code>	<code>dapl</code>
<code>rdssm</code>	<code>shm:dapl</code>
<code>&lt;provider&gt;</code>	Optional DAPL* provider name (only for the <code>rdma</code> and the <code>rdssm</code> devices)  <code>I_MPI_DAPL_PROVIDER=&lt;provider&gt;</code> or <code>I_MPI_DAPL_UD_PROVIDER=&lt;provider&gt;</code>

Use the `<provider>` specification only for the `{rdma, rdssm}` devices.

For example, to select the OFED\* InfiniBand\* device, use the following command:

```
$ mpiexec -n <# of processes> \
    -env I_MPI_DEVICE rdssm:OpenIB-cma <executable>
```

For these devices, if *<provider>* is not specified, the first DAPL\* provider in the `/etc/dat.conf` file is used.

## Description

Set this environment variable to select a specific fabric combination. If the requested fabric(s) is not available, Intel® MPI Library can fall back to other fabric(s). See [I\\_MPI\\_FALLBACK](#) for details. If the `I_MPI_FABRICS` environment variable is not defined, Intel® MPI Library selects the most appropriate fabric combination automatically.

The exact combination of fabrics depends on the number of processes started per node.

- If all processes start on one node, the library uses `shm` intra-node communication.
- If the number of started processes is less than or equal to the number of available nodes, the library uses the first available fabric from the fabrics list for inter-nodes communication.
- For other cases, the library uses `shm` for intra-node communication, and the first available fabric from the fabrics list for inter-nodes communication. See [I\\_MPI\\_FABRICS\\_LIST](#) for details.

The `shm` fabric is available for both Intel® and non-Intel microprocessors, but it may perform additional optimizations for Intel microprocessors than it performs for non-Intel microprocessors.

**NOTE:** The combination of selected fabrics ensures that the job runs, but this combination may not provide the highest possible performance for the given cluster configuration.

For example, to select shared-memory as the chosen fabric, use the following command:

```
$ mpiexec -n <# of processes> -env I_MPI_FABRICS shm <executable>
```

To select shared-memory and DAPL-capable network fabric as the chosen fabric combination, use the following command:

```
$ mpiexec -n <# of processes> -env I_MPI_FABRICS shm:dapl <executable>
```

To enable Intel® MPI Library to select most appropriate fabric combination automatically, use the following command:

```
$ mpiexec -n <# of procs> -perhost <# of procs per host> <executable>
```

Set the level of debug information to 2 or higher to check which fabrics have been initialized.

See [I\\_MPI\\_DEBUG](#) for details. For example:

```
[0] MPI startup(): shm and dapl data transfer modes
```

or

```
[0] MPI startup(): tcp data transfer mode
```

**NOTE:** If the `I_MPI_FABRICS` environment variable and the `I_MPI_DEVICE` environment variable are set at the same level (command line, environment, configuration files), the `I_MPI_FABRICS` environment variable has higher priority than the `I_MPI_DEVICE` environment variable.

## I\_MPI\_FABRICS\_LIST

Define a fabrics list.

**Syntax**

`I_MPI_FABRICS_LIST=<fabrics list>`

Where `<fabrics list> := <fabric>,...,<fabric>`

`<fabric> := {dapl, tcp, tmi, ofa}`

**Arguments**

<code>&lt;fabrics list&gt;</code>	Specify a fabrics list
<code>dapl, ofa, tcp, tmi</code>	This is the default value
<code>dapl, tcp, ofa, tmi</code>	If you specify <code>I_MPI_WAIT_MODE=enable</code> , this is the default value

**Description**

Set this environment variable to define a list of fabrics. The library uses the fabrics list to choose the most appropriate fabrics combination automatically. For more information on fabric combination, see [I\\_MPI\\_FABRICS](#)

For example, if `I_MPI_FABRICS_LIST=dapl, tcp`, `I_MPI_FABRICS` is not defined, and the initialization of DAPL-capable network fabrics fails, the library falls back to TCP-capable network fabric. For more information on fallback, see [I\\_MPI\\_FALLBACK](#).

**I\_MPI\_FALLBACK****(I\_MPI\_FALLBACK\_DEVICE)**

Set this environment variable to enable fallback to the first available fabric.

**Syntax**

`I_MPI_FALLBACK=<arg>`

**Deprecated Syntax**

`I_MPI_FALLBACK_DEVICE=<arg>`

**Arguments**

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Fall back to the first available fabric. This is the default value if <code>I_MPI_FABRICS(I_MPI_DEVICE)</code> environment variable is not set
<code>disable   no   off   0</code>	Terminate the job if MPI cannot initialize the one of the fabrics selected by the <code>I_MPI_FABRICS</code> environment variable. This is the default value if you set <code>I_MPI_FABRICS(I_MPI_DEVICE)</code> environment variable

**Description**

Set this environment variable to control fallback to the first available fabric.

If `I_MPI_FALLBACK` is set to `enable` and an attempt to initialize a specified fabric fails, the library uses the first available fabric from the list of fabrics. See [I\\_MPI\\_FABRICS\\_LIST](#) for details.

If `I_MPI_FALLBACK` is set to `disable` and an attempt to initialize a specified fabric fails, the library terminates the MPI job.

**NOTE:** If `I_MPI_FABRICS` is set and `I_MPI_FALLBACK=enable`, the library falls back to fabrics with higher numbers in the fabrics list. For example, if `I_MPI_FABRICS=dapl`, `I_MPI_FABRICS_LIST=ofa, tmi, dapl, tcp`, `I_MPI_FALLBACK=enable` and the

initialization of DAPL-capable network fabrics fails, the library falls back to TCP-capable network fabric.

## I\_MPI\_EAGER\_THRESHOLD

Change the eager/rendezvous message size threshold for all devices.

### Syntax

`I_MPI_EAGER_THRESHOLD=<nbytes>`

### Arguments

<code>&lt;nbytes&gt;</code>	Set the eager/rendezvous message size threshold
<code>&gt; 0</code>	The default <code>&lt;nbytes&gt;</code> value is equal to <b>262144</b> bytes

### Description

Set this environment variable to control the protocol used for point-to-point communication:

- Messages shorter than or equal in size to `<nbytes>` are sent using the eager protocol.
- Messages larger than `<nbytes>` are sent using the rendezvous protocol. The rendezvous protocol uses memory more efficiently.

## I\_MPI\_INTRANODE\_EAGER\_THRESHOLD

Change the eager/rendezvous message size threshold for intra-node communication mode.

### Syntax

`I_MPI_INTRANODE_EAGER_THRESHOLD=<nbytes>`

### Arguments

<code>&lt;nbytes&gt;</code>	Set the eager/rendezvous message size threshold for intra-node communication
<code>&gt; 0</code>	The default <code>&lt;nbytes&gt;</code> value is equal to <b>262144</b> bytes for all fabrics except <code>shm</code> . For <code>shm</code> , cutover point is equal to the value of <code>I_MPI_SHM_CELL_SIZE</code> environment variable

### Description

Set this environment variable to change the protocol used for communication within the node:

- Messages shorter than or equal in size to `<nbytes>` are sent using the eager protocol.
- Messages larger than `<nbytes>` are sent using the rendezvous protocol. The rendezvous protocol uses the memory more efficiently.

If `I_MPI_INTRANODE_EAGER_THRESHOLD` is not set, the value of `I_MPI_EAGER_THRESHOLD` is used.

## I\_MPI\_INTRANODE\_DIRECT\_COPY

Turn on/off the intranode direct copy communication mode.

### Syntax

`I_MPI_INTRANODE_DIRECT_COPY=<arg>`

### Arguments

<code>&lt;arg&gt;</code>	Binary indicator
--------------------------	------------------

<code>enable   yes   on   1</code>	Turn on the direct copy communication mode
<code>disable   no   off   0</code>	Turn off the direct copy communication mode. This is the default value

### Description

Set this environment variable to specify the communication mode within the node. If the direct copy communication mode is enabled, data transfer algorithms are selected according to the following scheme:

- Messages shorter than or equal to the threshold value of the `I_MPI_INTRANODE_EAGER_THRESHOLD` environment variable are transferred using the shared memory.
- Messages larger than the threshold value of the `I_MPI_INTRANODE_EAGER_THRESHOLD` environment variable are transferred through the direct process memory access.

## I\_MPI\_SPIN\_COUNT

Control the spin count value.

### Syntax

`I_MPI_SPIN_COUNT=<scount>`

### Arguments

<code>&lt;scount&gt;</code>	Define the loop spin count when polling fabric(s)
<code>&gt; 0</code>	The default <code>&lt;scount&gt;</code> value is equal to <code>1</code> when more than one process runs per processor/core. Otherwise the value equals <code>250</code>

### Description

Set the spin count limit. The loop for polling the fabric(s) spins `<scount>` times before freeing the processes if no incoming messages are received for processing. Smaller values for `<scount>` cause the Intel® MPI Library to release the processor more frequently.

Use the `I_MPI_SPIN_COUNT` environment variable for tuning application performance. The best value for `<scount>` can be chosen on an experimental basis. It depends on the particular computational environment and application.

## I\_MPI\_SCALABLE\_OPTIMIZATION

### (I\_MPI SOCK\_SCALABLE\_OPTIMIZATION)

Turn on/off scalable optimization of the network fabric communication.

### Syntax

`I_MPI_SCALABLE_OPTIMIZATION=<arg>`

### Deprecated Syntax

`I_MPI_SOCKET_SCALABLE_OPTIMIZATION=<arg>`

### Arguments

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Turn on scalable optimization of the network fabric communication. This is the default for 16 or more processes
<code>disable   no   off   0</code>	Turn off scalable optimization of the network fabric communication.

	This is the default for less than 16 processes
--	--

Description

Set this environment variable to enable scalable optimization of the network fabric communication. In most cases, using optimization decreases latency and increases bandwidth for a large number of processes.

**NOTE:** Old notification `I_MPI SOCK_SCALABLE_OPTIMIZATION` is equal to `I_MPI_SCALABLE_OPTIMIZATION` for `tcp` fabric.

I\_MPI\_WAIT\_MODE

Turn on/off wait mode.

Syntax

`I_MPI_WAIT_MODE=<arg>`

Arguments

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Turn on the wait mode
<code>disable   no   off   0</code>	Turn off the wait mode. This is the default

Description

Set this environment variable to control the wait mode. If this mode is enabled, the processes wait for receiving messages without polling the fabric(s). This mode can save CPU time for other tasks.

Use the Native POSIX Thread Library\* with the wait mode for `shm` communications.

**NOTE:** To check which version of the thread library is installed, use the following command:

`$ getconf GNU_LIBPTHREAD_VERSION`

I\_MPI\_DYNAMIC\_CONNECTION

(I\_MPI\_USE\_DYNAMIC\_CONNECTIONS)

Turn on/off the dynamic connection establishment.

Syntax

`I_MPI_DYNAMIC_CONNECTION=<arg>`

Deprecated Syntax

`I_MPI_USE_DYNAMIC_CONNECTIONS=<arg>`

Arguments

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Turn on the dynamic connection establishment. This is the default for 64 or more processes
<code>disable   no   off   0</code>	Turn off the dynamic connection establishment. This is the default for less than 64 processes

Description

Set this environment variable to control dynamic connection establishment.

- If this mode is enabled, all connections are established at the time of the first communication between each pair of processes.
- If this mode is disabled, all connections are established upfront.

The default value depends on a number of processes in the MPI job. The dynamic connection establishment is off if a total number of processes is less than 64.

## 3.3.2 Shared Memory Control

### I\_MPI\_SHM\_CACHE\_BYPASS

#### (I\_MPI\_CACHE\_BYPASS)

Control the message transfer algorithm for the shared memory.

#### Syntax

`I_MPI_SHM_CACHE_BYPASS=<arg>`

#### Deprecated Syntax

`I_MPI_CACHE_BYPASS=<arg>`

#### Arguments

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Enable message transfer bypass cache. This is the default value
<code>disable   no   off   0</code>	Disable message transfer bypass cache

#### Description

Set this environment variable to enable/disable message transfer bypass cache for the shared memory. When enabled, the MPI sends the messages greater than or equal in size to the value specified by the `I_MPI_SHM_CACHE_BYPASS_THRESHOLD` environment variable through the bypass cache. This feature is enabled by default.

### I\_MPI\_SHM\_CACHE\_BYPASS\_THRESHOLDS

#### (I\_MPI\_CACHE\_BYPASS\_THRESHOLDS)

Set the messages copying algorithm threshold.

#### Syntax

`I_MPI_SHM_CACHE_BYPASS_THRESHOLDS=<nb_send>, <nb_recv>[ , <nb_send_pk>, <nb_recv_pk>]`

#### Deprecated Syntax

`I_MPI_CACHE_BYPASS_THRESHOLDS=<nb_send>, <nb_recv>[ , <nb_send_pk>, <nb_recv_pk>]`

#### Arguments

<code>&lt;nb_send&gt;</code>	Set the threshold for sent messages in the following situations: <ul style="list-style-type: none"> <li>• Processes are pinned on cores that are not located in the same physical processor package</li> <li>• Processes are not pinned</li> </ul>
<code>&gt;= 0</code>	<ul style="list-style-type: none"> <li>• For machines optimized with Intel® Streaming SIMD Extensions 4.2 (Intel® SSE4.2) or Intel® AES New Instructions (Intel® AES-</li> </ul>

	<p>NI), the default <code>&lt;nb_send&gt;</code> value is <code>-1</code>. This value disables the copying bypass cache</p> <ul style="list-style-type: none"> <li>For other architectures, the default <code>&lt;nb_send&gt;</code> value is <code>16,384</code> bytes</li> </ul>
<code>&lt;nb_recv&gt;</code>	<p>Set the threshold for received messages in the following situations:</p> <ul style="list-style-type: none"> <li>Processes are pinned on cores that are not located in the same physical processor package</li> <li>Processes are not pinned</li> </ul>
<code>&gt;= 0</code>	<ul style="list-style-type: none"> <li>For machines optimized with Intel® SSE4.2, the default <code>&lt;nb_send&gt;</code> value is <code>-1</code>. This value disables the copying bypass cache</li> <li>For machines optimized with Intel® AES-NI, the default <code>&lt;nb_send&gt;</code> value is <code>MAX(1Mb, L3/NP)</code>, where <code>L3</code> indicates the size of Level 3 cache and <code>NP</code> indicates the number of processes on the node</li> <li>For other architectures, the default <code>&lt;nb_recv_pk&gt;</code> value is <code>2,097,152</code> bytes</li> </ul>
<code>&lt;nb_send_pk&gt;</code>	<p>Set the threshold for sent messages when processes are pinned on cores located in the same physical processor package</p>
<code>&gt;= 0</code>	<p>The default <code>&lt;nb_send_pk&gt;</code> value is <code>-1</code> (copying bypass cache is disabled)</p>
<code>&lt;nb_recv_pk&gt;</code>	<p>Set the threshold for received messages when processes are pinned on cores located in the same physical processor package</p>
<code>&gt;= 0</code>	<ul style="list-style-type: none"> <li>For machines optimized with Intel® SSE4.2, the default <code>&lt;nb_send&gt;</code> value is <code>-1</code>. This value disables the copying bypass cache</li> <li>For machines optimized with Intel® AES-NI, the default <code>&lt;nb_send&gt;</code> value is <code>MAX(1Mb, L3/NP)</code>, where <code>L3</code> indicates the size of Level 3 cache and <code>NP</code> indicates the number of processes on the node</li> <li>For other architectures, the default <code>&lt;nb_recv_pk&gt;</code> value is <code>2,097,152</code> bytes</li> </ul>

### Description

Set this environment variable to control the thresholds for the message copying algorithm. MPI copies messages greater than or equal in size to the defined threshold values so that the messages bypass the cache. The value of `-1` disables cache bypass. This environment variable is valid only when `I_MPI_SHM_CACHE_BYPASS` is enabled.

This environment variable is available for both Intel and non-Intel microprocessors, but it may perform additional optimizations for Intel microprocessors than it performs for non-Intel microprocessors.

## I\_MPI\_SHM\_FBOX

Control the usage of the shared memory fast-boxes.

### Syntax

`I_MPI_SHM_FBOX=<arg>`

### Arguments

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Turn on the usage of fast-boxes. This is the default value
<code>disable   no   off   0</code>	Turn off the usage of fast-boxes

### Description

Set this environment variable to control the usage of fast-boxes. Each pair of MPI processes on the same computing node has two shared memory fast-boxes, for sending and receiving eager messages.

Turn off the usage of fast-boxes to avoid the overhead of message synchronization when the application uses mass transfer of short non-blocking messages.

## I\_MPI\_SHM\_FBOX\_SIZE

Set the size of the shared memory fastbox.

### Syntax

`I_MPI_SHM_FBOX_SIZE=<nbytes>`

### Arguments

<code>&lt;nbytes&gt;</code>	Size of shared memory fastbox in bytes
<code>&gt; 0</code>	The default <code>&lt;nbytes&gt;</code> value is equal to 65,472 bytes

### Description

Set this environment variable to define the size of shared memory fast-boxes. The value must be multiple of 64.

## I\_MPI\_SHM\_CELL\_NUM

Change the number of cells in the shared memory receiving queue.

### Syntax

`I_MPI_SHM_CELL_NUM=<num>`

### Arguments

<code>&lt;num&gt;</code>	The number of shared memory cells
<code>&gt; 0</code>	The default value is 128

### Description

Set this environment variable to define the number of cells in the shared memory receive queue. Each MPI process has own shared memory receive queue, where other processes put eager messages. The queue is used when shared memory fast-boxes are blocked by another MPI request.

## I\_MPI\_SHM\_CELL\_SIZE

Change the size of shared memory cell.

Syntax

```
I_MPI_SHM_CELL_SIZE=<nbytes>
```

Arguments

<nbytes>	Size of shared memory cell in bytes
> 0	The default <nbytes> value is equal to 65,472 bytes

Description

Set this environment variable to define the size of shared memory cells. The value must be a multiple of 64.

If a value is set, `I_MPI_INTRANODE_EAGER_THRESHOLD` is also changed and becomes equal to the given value.

I\_MPI\_SHM\_LMT

Control the usage of large message transfer (LMT) mechanism for the shared memory.

Syntax

```
I_MPI_SHM_LMT=<arg>
```

Deprecated Syntax

```
I_MPI_INTRANODE_DIRECT_COPY=<arg>
```

Arguments

<arg>	Binary indicator
shm	Turn on the usage of shared memory copy LMT mechanism. This is the default value on Linux OS*
direct	Turn on the usage of direct copy LMT mechanism. This is the default value on Windows OS*
disable   no   off   0	Turn off the usage of LMT mechanism

Description

Set this environment variable to control the usage of the large message transfer (LMT) mechanism. To transfer rendezvous messages, you can use the LMT mechanism by employing either of the following implementations:

- Use intermediate shared memory queues to send messages.
- Use direct copy mechanism that transfers messages without intermediate buffer.

**NOTE:** Two arguments of the `I_MPI_SHM_LMT` environment variable are related to the `I_MPI_INTRANODE_DIRECT_COPY` environment variable:

- `I_MPI_SHM_LMT=direct` is equal to the deprecated setting `I_MPI_INTRANODE_DIRECT_COPY=enable`.
- `I_MPI_SHM_LMT=shm` is equal to the deprecated setting `I_MPI_INTRANODE_DIRECT_COPY=disable`.

I\_MPI\_SHM\_LMT\_BUFFER\_NUM

(I\_MPI\_SHM\_NUM\_BUFFERS)

Change the number of shared memory buffers for the large message transfer (LMT) mechanism.

**Syntax**`I_MPI_SHM_LMT_BUFFER_NUM=<num>`**Deprecated Syntax**`I_MPI_SHM_NUM_BUFFERS= <num>`**Arguments**

<code>&lt;num&gt;</code>	The number of shared memory buffers for each process pair
<code>&gt; 0</code>	The default value is 8

**Description**

Set this environment variable to define the number of shared memory buffers between each process pair.

**I\_MPI\_SHM\_LMT\_BUFFER\_SIZE****(I\_MPI\_SHM\_BUFFER\_SIZE)**

Change the size of shared memory buffers for the LMT mechanism.

**Syntax**`I_MPI_SHM_LMT_BUFFER_SIZE=<nbytes>`**Deprecated Syntax**`I_MPI_SHM_BUFFER_SIZE=<nbytes>`**Arguments**

<code>&lt;nbytes&gt;</code>	The size of shared memory buffers in bytes
<code>&gt; 0</code>	The default <code>&lt;nbytes&gt;</code> value is equal to 32,768 bytes

**Description**

Set this environment variable to define the size of shared memory buffers for each pair of processes.

**I\_MPI\_SSHM**

Control the usage of the scalable shared memory mechanism.

**Syntax**`I_MPI_SSHM = <arg>`**Arguments**

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Turn on the usage of this mechanism
<code>disable   no   off   0</code>	Turn off the usage of this mechanism. This is the default value

**Description**

Set this environment variable to control the usage of an alternative shared memory mechanism. This mechanism replaces the shared memory fast-boxes, receive queues and LMT mechanism.

If a value is set, the `I_MPI_INTRANODE_EAGER_THRESHOLD` environment variable is changed and becomes equal to 262,144 bytes.

## I\_MPI\_SSHM\_BUFFER\_NUM

Change the number of shared memory buffers for the alternative shared memory mechanism.

### Syntax

`I_MPI_SSHM_BUFFER_NUM=<num>`

### Arguments

<code>&lt;num&gt;</code>	The number of shared memory buffers for each process pair
<code>&gt; 0</code>	The default value is <code>4</code>

### Description

Set this environment variable to define the number of shared memory buffers between each process pair.

## I\_MPI\_SSHM\_BUFFER\_SIZE

Change the size of shared memory buffers for the alternative shared memory mechanism.

### Syntax

`I_MPI_SSHM_BUFFER_SIZE=<nbytes>`

### Arguments

<code>&lt;nbytes&gt;</code>	The size of shared memory buffers in bytes
<code>&gt; 0</code>	The default <code>&lt;nbytes&gt;</code> value is <code>65,472</code> bytes

### Description

Set this environment variable to define the size of shared memory buffers for each pair of processes.

## I\_MPI\_SSHM\_DYNAMIC\_CONNECTION

Control the dynamic connection establishment for the alternative shared memory mechanism.

### Syntax

`I_MPI_SSHM_DYNAMIC_CONNECTION=<arg>`

### Arguments

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Turn on the dynamic connection establishment
<code>disable   no   off   0</code>	Turn off the dynamic connection establishment. This is the default value

### Description

Set this environment variable to control the dynamic connection establishment.

- If this mode is enabled, all connections are established at the time of the first communication between each pair of processes.
- If this mode is disabled, all connections are established upfront.

## I\_MPI\_SHM\_BYPASS

### (I\_MPI\_INTRANODE\_SHMEM\_BYPASS, I\_MPI\_USE\_DAPL\_INTRANODE)

Turn on/off the intra-node communication mode through network fabric along with `shm`.

#### Syntax

`I_MPI_SHM_BYPASS=<arg>`

#### Deprecated Syntaxes

`I_MPI_INTRANODE_SHMEM_BYPASS=<arg>`

`I_MPI_USE_DAPL_INTRANODE=<arg>`

#### Arguments

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Turn on the intra-node communication through network fabric
<code>disable   no   off   0</code>	Turn off the intra-node communication through network fabric. This is the default

#### Description

Set this environment variable to specify the communication mode within the node. If the intra-node communication mode through network fabric is enabled, data transfer algorithms are selected according to the following scheme:

- Messages shorter than or equal in size to the threshold value of the `I_MPI_INTRANODE_EAGER_THRESHOLD` environment variable are transferred using shared memory.
- Messages larger than the threshold value of the `I_MPI_INTRANODE_EAGER_THRESHOLD` environment variable are transferred through the network fabric layer.

**NOTE:** This environment variable is applicable only when shared memory and a network fabric are turned on either by default or by setting the `I_MPI_FABRICS` environment variable to `shm:<fabric>` or an equivalent `I_MPI_DEVICE` setting. This mode is available only for `dapl` and `tcp` fabrics.

## I\_MPI\_SHM\_SPIN\_COUNT

Control the spin count value for the shared memory fabric.

#### Syntax

`I_MPI_SHM_SPIN_COUNT=<scout>`

#### Arguments

<code>&lt;scout&gt;</code>	Define the spin count of the loop when polling the <code>shm</code> fabric
<code>&gt; 0</code>	When internode communication uses the <code>dapl</code> or <code>tcp</code> fabric, the default <code>&lt;scout&gt;</code> value is equal to 100 spins When internode communication uses the <code>ofa</code> , <code>tmi</code> or <code>dapl</code> (DAPL UD-enabled only) fabric, the default <code>&lt;scout&gt;</code> value is equal to 10 spins

#### Description

Set the spin count limit of the shared memory fabric to increase the frequency of polling. This configuration allows polling of the `shm` fabric `<scout>` times before the control is passed to the

overall network fabric polling mechanism. See [I\\_MPI\\_SPIN\\_COUNT](#) for details on higher level fabrics polling.

To tune application performance, use the `I_MPI_SHM_SPIN_COUNT` environment variable. You can choose the best value for `<scount>` on an experimental basis. It depends largely on the application and the particular computation environment. An increase in the `<scount>` value will benefit multi-core platforms when the application uses topological algorithms for message passing.

### 3.3.3 DAPL-capable Network Fabrics Control

#### I\_MPI\_DAPL\_PROVIDER

Define the DAPL provider to load.

##### Syntax

`I_MPI_DAPL_PROVIDER=<name>`

##### Arguments

<code>&lt;name&gt;</code>	Define the name of DAPL provider to load
---------------------------	--

##### Description

Set this environment variable to define the name of DAPL provider to load. This name is also defined in the `dat.conf` configuration file.

#### I\_MPI\_DAT\_LIBRARY

Select the DAT library to be used for DAPL\* provider.

##### Syntax

`I_MPI_DAT_LIBRARY=<library>`

##### Arguments

<code>&lt;library&gt;</code>	Specify the DAT library for DAPL provider to be used. Default values are <code>libdat.so</code> or <code>libdat.so.1</code> for DAPL* 1.2 providers and <code>libdat2.so</code> or <code>libdat2.so.2</code> for DAPL* 2.0 providers
------------------------------	--

##### Description

Set this environment variable to select a specific DAT library to be used for DAPL provider. If the library is not located in the dynamic loader search path, specify the full path to the DAT library. This environment variable affects only on DAPL and DAPL UD capable fabrics.

#### I\_MPI\_DAPL\_TRANSLATION\_CACHE

#### (I\_MPI\_RDMA\_TRANSLATION\_CACHE)

Turn on/off the memory registration cache in the DAPL path.

##### Syntax

`I_MPI_DAPL_TRANSLATION_CACHE=<arg>`

##### Deprecated Syntax

`I_MPI_RDMA_TRANSLATION_CACHE=<arg>`

##### Arguments

<code>&lt;arg&gt;</code>	Binary indicator
--------------------------	------------------

<code>enable   yes   on   1</code>	Turn on the memory registration cache. This is the default
<code>disable   no   off   0</code>	Turn off the memory registration cache

### Description

Set this environment variable to turn on/off the memory registration cache in the DAPL path.

The cache substantially increases performance, but may lead to correctness issues in certain rare situations. See product *Release Notes* for further details.

## I\_MPI\_DAPL\_TRANSLATION\_CACHE\_AVL\_TREE

Enable/disable the AVL tree\* based implementation of the RDMA translation cache in the DAPL path.

### Syntax

`I_MPI_DAPL_TRANSLATION_CACHE_AVL_TREE=<arg>`

### Arguments

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Turn on the AVL tree based RDMA translation cache
<code>disable   no   off   0</code>	Turn off the AVL tree based RDMA translation cache. This is the default value

### Description

Set this environment variable to enable the AVL tree based implementation of RDMA translation cache in the DAPL path. When the search in RDMA translation cache handles over 10,000 elements, the AVL tree based RDMA translation cache is faster than the default implementation.

## I\_MPI\_DAPL\_DIRECT\_COPY\_THRESHOLD

### (I\_MPI\_RDMA\_EAGER\_THRESHOLD, RDMA\_IBA\_EAGER\_THRESHOLD)

Change the threshold of the DAPL direct-copy protocol.

### Syntax

`I_MPI_DAPL_DIRECT_COPY_THRESHOLD=<nbytes>`

### Deprecated Syntaxes

`I_MPI_RDMA_EAGER_THRESHOLD=<nbytes>`

`RDMA_IBA_EAGER_THRESHOLD=<nbytes>`

### Arguments

<code>&lt;nbytes&gt;</code>	Define the DAPL direct-copy protocol threshold
<code>&gt; 0</code>	The default <code>&lt;nbytes&gt;</code> value is equal to <b>16456</b> bytes

### Description

Set this environment variable to control the DAPL direct-copy protocol threshold. Data transfer algorithms for the DAPL-capable network fabrics are selected based on the following scheme:

- Messages shorter than or equal to `<nbytes>` are sent using the eager protocol through the internal pre-registered buffers. This approach is faster for short messages.

- Messages larger than *<nbytes>* are sent using the direct-copy protocol. It does not use any buffering but involves registration of memory on sender and receiver sides. This approach is faster for large messages.

This environment variable is available for both Intel® and non-Intel microprocessors, but it may perform additional optimizations for Intel microprocessors than it performs for non-Intel microprocessors.

## I\_MPI\_DAPL\_EAGER\_MESSAGE\_AGGREGATION

Control the use of concatenation for adjourned MPI send requests. Adjourned MPI send requests are those that cannot be sent immediately.

### Syntax

`I_MPI_DAPL_EAGER_MESSAGE_AGGREGATION = <arg>`

### Arguments

<i>&lt;arg&gt;</i>	Binary indicator
<code>enable   yes   on   1</code>	Enable the concatenation for adjourned MPI send requests
<code>disable   no   off   0</code>	Disable the concatenation for adjourned MPI send requests. This is the default value

Set this environment variable to control the use of concatenation for adjourned MPI send requests intended for the same MPI rank. In some cases, this mode can improve the performance of applications, especially when `MPI_Isend()` is used with short message sizes and the same destination rank, such as:

```
for( i = 0; i< NMSG; i++)
    {ret = MPI_Isend( sbuf[i], MSG_SIZE, datatype, dest , tag, \
        comm, &req_send[i]);
    }
```

## I\_MPI\_DAPL\_DYNAMIC\_CONNECTION\_MODE

### (I\_MPI\_DYNAMIC\_CONNECTION\_MODE, I\_MPI\_DYNAMIC\_CONNECTIONS\_MODE)

Choose the algorithm for establishing the DAPL\* connections.

### Syntax

`I_MPI_DAPL_DYNAMIC_CONNECTION_MODE= <arg>`

### Deprecated Syntax

`I_MPI_DYNAMIC_CONNECTION_MODE= <arg>`

`I_MPI_DYNAMIC_CONNECTIONS_MODE= <arg>`

### Arguments

<i>&lt;arg&gt;</i>	Mode selector
<code>reject</code>	Deny one of the two simultaneous connection requests. This is the default
<code>disconnect</code>	Deny one of the two simultaneous connection requests after both connections have been established

Description

Set this environment variable to choose the algorithm for handling dynamically established connections for DAPL-capable fabrics according to the following scheme:

- In the `reject` mode, if two processes initiate the connection simultaneously, one of the requests is rejected.
- In the `disconnect` mode, both connections are established, but then one is disconnected. The `disconnect` mode is provided to avoid a bug in certain DAPL\* providers.

I\_MPI\_DAPL\_SCALABLE\_PROGRESS

(I\_MPI\_RDMA\_SCALABLE\_PROGRESS)

Turn on/off scalable algorithm for DAPL read progress.

Syntax

I\_MPI\_DAPL\_SCALABLE\_PROGRESS=<arg>

Deprecated Syntax

I\_MPI\_RDMA\_SCALABLE\_PROGRESS= <arg>

Arguments

<arg>	Binary indicator
enable   yes   on   1	Turn on scalable algorithm. When the number of processes is larger than 128, this is the default value
disable   no   off   0	Turn off scalable algorithm. When the number of processes is less than or equal to 128, this is the default value

Description

Set this environment variable to enable scalable algorithm for the DAPL read progress. In some cases, this provides advantages for systems with many processes.

I\_MPI\_DAPL\_BUFFER\_NUM

(I\_MPI\_RDMA\_BUFFER\_NUM, NUM\_RDMA\_BUFFER)

Change the number of internal pre-registered buffers for each process pair in the DAPL path.

Syntax

I\_MPI\_DAPL\_BUFFER\_NUM= <nbuf>

Deprecated Syntaxes

I\_MPI\_RDMA\_BUFFER\_NUM= <nbuf>

NUM\_RDMA\_BUFFER= <nbuf>

Arguments

<nbuf>	Define the number of buffers for each pair in a process group
> 0	The default value is 16

Description

Set this environment variable to change the number of the internal pre-registered buffers for each process pair in the DAPL path.

**NOTE:** The more pre-registered buffers are available, the more memory is used for every established connection.

## I\_MPI\_DAPL\_BUFFER\_SIZE

### (I\_MPI\_RDMA\_BUFFER\_SIZE, I\_MPI\_RDMA\_VBUF\_TOTAL\_SIZE)

Change the size of internal pre-registered buffers for each process pair in the DAPL path.

#### Syntax

`I_MPI_DAPL_BUFFER_SIZE=<nbytes>`

#### Deprecated Syntaxes

`I_MPI_RDMA_BUFFER_SIZE=<nbytes>`

`I_MPI_RDMA_VBUF_TOTAL_SIZE=<nbytes>`

#### Arguments

<code>&lt;nbytes&gt;</code>	Define the size of pre-registered buffers
<code>&gt; 0</code>	The default <code>&lt;nbytes&gt;</code> value is equal to 16,640 bytes

#### Description

Set this environment variable to define the size of the internal pre-registered buffer for each process pair in the DAPL path. The actual size is calculated by adjusting the `<nbytes>` to align the buffer to an optimal value.

## I\_MPI\_DAPL\_RNDV\_BUFFER\_ALIGNMENT

### (I\_MPI\_RDMA\_RNDV\_BUFFER\_ALIGNMENT, I\_MPI\_RDMA\_RNDV\_BUF\_ALIGN)

Define the alignment of the sending buffer for the DAPL direct-copy transfers.

#### Syntax

`I_MPI_DAPL_RNDV_BUFFER_ALIGNMENT=<arg>`

#### Deprecated Syntaxes

`I_MPI_RDMA_RNDV_BUFFER_ALIGNMENT=<arg>`

`I_MPI_RDMA_RNDV_BUF_ALIGN=<arg>`

#### Arguments

<code>&lt;arg&gt;</code>	Define the alignment for the sending buffer
<code>&gt; 0 and a power of 2</code>	The default value is 128

Set this environment variable to define the alignment of the sending buffer for DAPL direct-copy transfers. When a buffer specified in a DAPL operation is aligned to an optimal value, the data transfer bandwidth may be increased.

## I\_MPI\_DAPL\_RDMA\_RNDV\_WRITE

### (I\_MPI\_RDMA\_RNDV\_WRITE, I\_MPI\_USE\_RENDEZVOUS\_RDMA\_WRITE)

Turn on/off the RDMA Write-based rendezvous direct-copy protocol in the DAPL path.

#### Syntax

`I_MPI_DAPL_RDMA_RNDV_WRITE=<arg>`

**Deprecated Syntaxes**`I_MPI_RDMA_RNDV_WRITE=<arg>``I_MPI_USE_RENDEZVOUS_RDMA_WRITE=<arg>`**Arguments**

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Turn on the RDMA Write rendezvous direct-copy protocol
<code>disable   no   off   0</code>	Turn off the RDMA Write rendezvous direct-copy protocol

**Description**

Set this environment variable to select the RDMA Write-based rendezvous direct-copy protocol in the DAPL path. Certain DAPL\* providers have a slow RDMA Read implementation on certain platforms. Switching on the rendezvous direct-copy protocol based on the RDMA Write operation can increase performance in these cases. The default value depends on the DAPL provider attributes.

**I\_MPI\_DAPL\_CHECK\_MAX\_RDMA\_SIZE****(I\_MPI\_RDMA\_CHECK\_MAX\_RDMA\_SIZE)**

Check the value of the DAPL attribute, `max_rdma_size`.

**Syntax**`I_MPI_DAPL_CHECK_MAX_RDMA_SIZE=<arg>`**Deprecated Syntax**`I_MPI_RDMA_CHECK_MAX_RDMA_SIZE=<arg>`**Arguments**

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Check the value of the DAPL* attribute <code>max_rdma_size</code>
<code>disable   no   off   0</code>	Do not check the value of the DAPL* attribute <code>max_rdma_size</code> . This is the default value

**Description**

Set this environment variable to control message fragmentation according to the following scheme:

- If this mode is enabled, the Intel® MPI Library fragmentizes the messages bigger than the value of the DAPL attribute `max_rdma_size`
- If this mode is disabled, the Intel® MPI Library does not take into account the value of the DAPL attribute `max_rdma_size` for message fragmentation

## I\_MPI\_DAPL\_MAX\_MSG\_SIZE

### (I\_MPI\_RDMA\_MAX\_MSG\_SIZE)

Control message fragmentation threshold.

#### Syntax

`I_MPI_DAPL_MAX_MSG_SIZE=<nbytes>`

#### Deprecated Syntax

`I_MPI_RDMA_MAX_MSG_SIZE=<nbytes>`

#### Arguments

<code>&lt;nbytes&gt;</code>	Define the maximum message size that can be sent through DAPL without fragmentation
<code>&gt; 0</code>	If the <code>I_MPI_DAPL_CHECK_MAX_RDMA_SIZE</code> environment variable is enabled, the default <code>&lt;nbytes&gt;</code> value is equal to the <code>max_rdma_size</code> DAPL attribute value. Otherwise the default value is <code>MAX_INT</code>

#### Description

Set this environment variable to control message fragmentation size according to the following scheme:

- If the `I_MPI_DAPL_CHECK_MAX_RDMA_SIZE` environment variable is set to `disable`, the Intel® MPI Library fragmentizes the messages whose sizes are greater than `<nbytes>`.
- If the `I_MPI_DAPL_CHECK_MAX_RDMA_SIZE` environment variable is set to `enable`, the Intel® MPI Library fragmentizes the messages whose sizes are greater than the minimum of `<nbytes>` and the `max_rdma_size` DAPL\* attribute value.

## I\_MPI\_DAPL\_CONN\_EVD\_SIZE

### (I\_MPI\_RDMA\_CONN\_EVD\_SIZE, I\_MPI\_CONN\_EVD\_QLEN)

Define the event queue size of the DAPL event dispatcher for connections.

#### Syntax

`I_MPI_DAPL_CONN_EVD_SIZE=<size>`

#### Deprecated Syntaxes

`I_MPI_RDMA_CONN_EVD_SIZE=<size>`

`I_MPI_CONN_EVD_QLEN=<size>`

#### Arguments

<code>&lt;size&gt;</code>	Define the length of the event queue
<code>&gt; 0</code>	The default value is <code>2*number of processes + 32</code> in the MPI job

#### Description

Set this environment variable to define the event queue size of the DAPL event dispatcher that handles connection related events. If this environment variable is set, the minimum value between `<size>` and the value obtained from the provider is used as the size of the event queue. The provider is required to supply a queue size that is at least equal to the calculated value, but it can also provide a larger queue size.

## I\_MPI\_DAPL\_SR\_THRESHOLD

Change the threshold of switching send/recv to `rdma` path for DAPL wait mode.

**Syntax**

`I_MPI_DAPL_SR_THRESHOLD=<arg>`

**Arguments**

<code>&lt;nbytes&gt;</code>	Define the message size threshold of switching send/recv to <code>rdma</code>
<code>&gt;= 0</code>	The default <code>&lt;nbytes&gt;</code> value is <code>256</code> bytes

**Description**

Set this environment variable to control the protocol used for point-to-point communication in DAPL wait mode:

- Messages shorter than or equal in size to `<nbytes>` are sent using DAPL send/recv data transfer operations.
- Messages greater in size than `<nbytes>` are sent using DAPL RDMA WRITE or RDMA WRITE immediate data transfer operations.

**I\_MPI\_DAPL\_SR\_BUF\_NUM**

Change the number of internal pre-registered buffers for each process pair used in DAPL wait mode for send/recv path.

**Syntax**

`I_MPI_DAPL_SR_BUF_NUM=<nbuf>`

**Arguments**

<code>&lt;nbuf&gt;</code>	Define the number of send/recv buffers for each pair in a process group
<code>&gt; 0</code>	The default value is <code>32</code>

**Description**

Set this environment variable to change the number of the internal send/recv pre-registered buffers for each process pair.

**I\_MPI\_DAPL\_RDMA\_WRITE\_IMM****(I\_MPI\_RDMA\_WRITE\_IMM)**

Enable/disable RDMA Write with immediate data InfiniBand (IB) extension in DAPL wait mode.

**Syntax**

`I_MPI_DAPL_RDMA_WRITE_IMM=<arg>`

**Deprecated syntax**

`I_MPI_RDMA_WRITE_IMM=<arg>`

**Arguments**

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Turn on RDMA Write with immediate data IB extension
<code>disable   no   off   0</code>	Turn off RDMA Write with immediate data IB extension

Description

Set this environment variable to utilize RDMA Write with immediate data IB extension. The algorithm is enabled if this environment variable is set and a certain DAPL provider attribute indicates that RDMA Write with immediate data IB extension is supported.

I\_MPI\_DAPL\_DESIRED\_STATIC\_CONNECTIONS\_NUM

Define the number of processes that establish DAPL static connections at the same time.

Syntax

I\_MPI\_DAPL\_DESIRED\_STATIC\_CONNECTIONS\_NUM=<num\_procesess>

Arguments

<num_procesess>	Define the number of processes that establish DAPL static connections at the same time
> 0	The default <num_procesess> value is equal to 256

Description

Set this environment variable to control the algorithm of DAPL static connection establishment.

If the number of processes in the MPI job is less than or equal to <num\_procesess>, all MPI processes establish the static connections simultaneously. Otherwise, the processes are distributed into several groups. The number of processes in each group is calculated to be close to <num\_procesess>. Then static connections are established in several iterations, including intergroup connection setup.

3.3.4 DAPL UD-capable Network Fabrics Control

I\_MPI\_DAPL\_UD

Enable/disable using DAPL UD path.

Syntax

I\_MPI\_DAPL\_UD=<arg>

Arguments

<arg>	Binary indicator
enable   yes   on   1	Turn on using DAPL UD IB extension
disable   no   off   0	Turn off using DAPL UD IB extension. This is the default value

Description

Set this environment variable to enable DAPL UD path for transferring data. The algorithm is enabled if you set this environment variable and a certain DAPL provider attribute indicates that UD IB extension is supported.

I\_MPI\_DAPL\_UD\_PROVIDER

Define the DAPL provider to work with IB UD transport.

Syntax

I\_MPI\_DAPL\_UD\_PROVIDER=<name>

## Arguments

<code>&lt;name&gt;</code>	Define the name of DAPL provider to load
---------------------------	--

## Description

Set this environment variable to define the name of DAPL provider to load. This name is also defined in the `dat.conf` configuration file. Make sure that specified DAPL provider supports UD IB extension.

## I\_MPI\_DAPL\_UD\_DIRECT\_COPY\_THRESHOLD

Change the message size threshold of the DAPL UD direct-copy protocol.

## Syntax

`I_MPI_DAPL_UD_DIRECT_COPY_THRESHOLD= <nbytes>`

## Arguments

<code>&lt;nbytes&gt;</code>	Define the DAPL UD direct-copy protocol threshold
<code>&gt; 0</code>	The default <code>&lt;nbytes&gt;</code> value is equal to <code>16456</code> bytes

## Description

Set this environment variable to control the DAPL UD direct-copy protocol threshold. Data transfer algorithms for the DAPL-capable network fabrics are selected based on the following scheme:

- Messages shorter than or equal to `<nbytes>` are sent using the eager protocol through the internal pre-registered buffers. This approach is faster for short messages.
- Messages larger than `<nbytes>` are sent using the direct-copy protocol. It does not use any buffering but involves registration of memory on sender and receiver sides. This approach is faster for large messages.

This environment variable is available for both Intel® and non-Intel microprocessors, but it may perform additional optimizations for Intel microprocessors than it performs for non-Intel microprocessors.

## I\_MPI\_DAPL\_UD\_RECV\_BUFFER\_NUM

Change the number of the internal pre-registered UD buffers for receiving messages.

## Syntax

`I_MPI_DAPL_UD_RECV_BUFFER_NUM= <nbuf>`

## Arguments

<code>&lt;nbuf&gt;</code>	Define the number of buffers for receiving messages
<code>&gt; 0</code>	The default value is <code>16 + n*4</code> where <code>n</code> is a total number of process in MPI job

## Description

Set this environment variable to change the number of the internal pre-registered buffers for receiving messages. These buffers are common for all connections or process pairs.

**NOTE:** The pre-registered buffers use up memory, however they help avoid the loss of packets.

## I\_MPI\_DAPL\_UD\_SEND\_BUFFER\_NUM

Change the number of internal pre-registered UD buffers for sending messages.

**Syntax**

`I_MPI_DAPL_UD_SEND_BUFFER_NUM=<nbuf>`

**Arguments**

<code>&lt;nbuf&gt;</code>	Define the number of buffers for sending messages
<code>&gt; 0</code>	The default value is <code>16 + n*4</code> where n is a total number of process in MPI job

**Description**

Set this environment variable to change the number of the internal pre-registered buffers for sending messages. These buffers are common for all connections or process pairs.

**NOTE:** The pre-registered buffers use up memory, however they help avoid the loss of packets.

**I\_MPI\_DAPL\_UD\_ACK\_SEND\_POOL\_SIZE**

Change the number of ACK UD buffers for sending messages.

**Syntax**

`I_MPI_DAPL_UD_ACK_SEND_POOL_SIZE=<nbuf>`

**Arguments**

<code>&lt;nbuf&gt;</code>	Define the number of ACK UD buffers for sending messages
<code>&gt; 0</code>	The default value is <code>256</code>

**Description**

Set this environment variable to change the number of the internal pre-registered ACK buffers for sending service messages. These buffers are common for all connections or process pairs.

**I\_MPI\_DAPL\_UD\_ACK\_RECV\_POOL\_SIZE**

Change the number of ACK UD buffers for receiving messages.

**Syntax**

`I_MPI_DAPL_UD_ACK_RECV_POOL_SIZE=<nbuf>`

**Arguments**

<code>&lt;nbuf&gt;</code>	Define the number of ACK UD buffers for receiving messages
<code>&gt; 0</code>	The default value is <code>512+n*4</code> , where n is total number of process in MPI job

**Description**

Set this environment variable to change the number of the internal pre-registered ACK buffers for receiving service messages. These buffers are common for all connections or process pairs.

**I\_MPI\_DAPL\_UD\_TRANSLATION\_CACHE**

Turn on/off the memory registration cache in the DAPL UD path.

**Syntax**

`I_MPI_DAPL_UD_TRANSLATION_CACHE=<arg>`

**Arguments**

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Turn on the memory registration cache. This is the default
<code>disable   no   off   0</code>	Turn off the memory registration cache

**Description**

Set this environment variable to turn off the memory registration cache in the DAPL UD path.

Using the cache substantially improves performance. See product *Release Notes* for further details.

**I\_MPI\_DAPL\_UD\_TRANSLATION\_CACHE\_AVL\_TREE**

Enable/disable the AVL\* tree based implementation of RDMA translation cache in the DAPL UD path.

**Syntax**

`I_MPI_DAPL_UD_TRANSLATION_CACHE_AVL_TREE=<arg>`

**Arguments**

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Turn on the AVL tree based RDMA translation cache
<code>disable   no   off   0</code>	Turn off the AVL tree based RDMA translation cache. This is the default value

**Description**

Set this environment variable to enable the AVL tree based implementation of RDMA translation cache in the DAPL UD path. When the search in RDMA translation cache handles over 10,000 elements, the AVL tree based RDMA translation cache is faster than the default implementation.

**I\_MPI\_DAPL\_UD\_REQ\_EVD\_SIZE**

Define the event queue size of the DAPL UD event dispatcher for sending data transfer operations.

**Syntax**

`I_MPI_DAPL_UD_REQ_EVD_SIZE=<size>`

**Arguments**

<code>&lt;size&gt;</code>	Define the length of the event queue
<code>&gt; 0</code>	The default value is <b>2,000</b>

**Description**

Set this environment variable to define the event queue size of the DAPL event dispatcher that handles completions of sending DAPL UD data transfer operations (DTO). If this environment variable is set, the minimum value between `<size>` and the value obtained from the provider is used as the size of the event queue. The provider is required to supply a queue size that is at least equal to the calculated value, but it can also provide a larger queue size.

**I\_MPI\_DAPL\_UD\_CONN\_EVD\_SIZE**

Define the event queue size of the DAPL UD event dispatcher for connections.

**Syntax**

`I_MPI_DAPL_UD_CONN_EVD_SIZE=<size>`

**Arguments**

<code>&lt;size&gt;</code>	Define the length of the event queue
<code>&gt; 0</code>	The default value is <code>2*number of processes + 32</code>

**Description**

Set this environment variable to define the event queue size of the DAPL event dispatcher that handles connection related events. If this environment variable is set, the minimum value between `<size>` and the value obtained from the provider is used as the size of the event queue. The provider is required to supply a queue size that is at least equal to the calculated value, but it can also provide a larger queue size.

**I\_MPI\_DAPL\_UD\_RECV\_EVD\_SIZE**

Define the event queue size of the DAPL UD event dispatcher for receiving data transfer operations.

**Syntax**

`I_MPI_DAPL_UD_RECV_EVD_SIZE=<size>`

**Arguments**

<code>&lt;size&gt;</code>	Define the length of the event queue
<code>&gt; 0</code>	The default value depends on the number UD and ACK buffers

**Description**

Set this environment variable to define the event queue size of the DAPL event dispatcher that handles completions of receiving DAPL UD data transfer operations (DTO). If this environment variable is set, the minimum value between `<size>` and the value obtained from the provider is used as the size of the event queue. The provider is required to supply a queue size that is at least equal to the calculated value, but it can also provide a larger queue size.

**I\_MPI\_DAPL\_UD\_RNDV\_MAX\_BLOCK\_LEN**

Define maximum size of block that is passed at one iteration of DAPL UD direct-copy protocol.

**Syntax**

`I_MPI_DAPL_UD_RNDV_MAX_BLOCK_LEN=<nbytes>`

**Arguments**

<code>&lt;arg&gt;</code>	Define maximum size of block that is passed at one iteration of DAPL UD direct-copy protocol
<code>&gt; 0</code>	The default value is <code>1,048,576</code>

Set this environment variable to define the maximum size of memory block that is passed at one iteration of DAPL UD direct-copy protocol. If the size of message in direct-copy protocol is greater than given value, the message will be divided in several blocks and passed in several operations.

**I\_MPI\_DAPL\_UD\_RNDV\_BUFFER\_ALIGNMENT**

Define the alignment of the sending buffer for the DAPL UD direct-copy transfers.

**Syntax**

`I_MPI_DAPL_UD_RNDV_BUFFER_ALIGNMENT=<arg>`

**Arguments**

<code>&lt;arg&gt;</code>	Define the alignment of the sending buffer
<code>&gt; 0 and a power of 2</code>	The default value is <code>16</code>

Set this environment variable to define the alignment of the sending buffer for DAPL direct-copy transfers. When a buffer specified in a DAPL operation is aligned to an optimal value, this may increase data transfer bandwidth.

**I\_MPI\_DAPL\_UD\_RNDV\_COPY\_ALIGNMENT\_THRESHOLD**

Define threshold where alignment is applied to send buffer for the DAPL UD direct-copy transfers.

**Syntax**

`I_MPI_DAPL_UD_RNDV_COPY_ALIGNMENT_THRESHOLD=<nbytes>`

**Arguments**

<code>&lt;nbytes&gt;</code>	Define send buffer alignment threshold
<code>&gt; 0 and a power of 2</code>	The default value is <code>32,768</code>

Set this environment variable to define the threshold where the alignment of the sending buffer is applied in DAPL direct-copy transfers. When a buffer specified in a DAPL operation is aligned to an optimal value, this may increase data transfer bandwidth.

**I\_MPI\_DAPL\_UD\_RNDV\_DYNAMIC\_CONNECTION**

Control the algorithm of dynamic connection establishment for DAPL UD endpoints used in the direct copy protocol.

**Syntax**

`I_MPI_DAPL_UD_RNDV_DYNAMIC_CONNECTION=<arg>`

**Arguments**

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Turns on the dynamic connection mode. This is the default value
<code>disable   no   off   0</code>	Turns off the dynamic connections mode

Set this variable to control the dynamic connection establishment of DAPL UD endpoints used in the direct copy protocol.

If you disable the dynamic connection mode, all possible connections are established during the MPI startup phase.

If you enable the mode, the connection is established when an application calls the MPI function to pass the data from one process to another and invokes the communication between the two processes.

**NOTE:** For the RNDV dynamic connection mode, the size of the messages passed in the data is larger than the value you set in the `I_MPI_DAPL_UD_DIRECT_COPY_THRESHOLD` environment variable.

## I\_MPI\_DAPL\_UD\_EAGER\_DYNAMIC\_CONNECTION

Control the algorithm of the dynamic connection establishment for DAPL UD endpoints used in eager protocol.

### Syntax

`I_MPI_DAPL_UD_EAGER_DYNAMIC_CONNECTION=<arg>`

### Arguments

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Turn on the dynamic connection mode. If the number of processes is over <code>64</code> , this is the default value
<code>disable   no   off   0</code>	Turn off the dynamic connections mode

Set this variable to control the dynamic connection establishment of DAPL UD endpoints involved in eager protocol. Eager protocol is used to transfer messages through internal pre-registered buffers.

If you disable this mode, all possible connections are established during MPI startup phase.

If you enable this mode, the connection is established when an application calls the MPI function to pass the data from one process to another and invokes the communication between the two processes.

**NOTE:** For the eager dynamic connection mode, the size of the messages passed in the data is shorter than or equal to the value you set in the `I_MPI_DAPL_UD_DIRECT_COPY_THRESHOLD` environment variable.

## I\_MPI\_DAPL\_UD\_DESIRED\_STATIC\_CONNECTIONS\_NUM

Define the number of processes that establish DAPL static connections at the same time.

### Syntax

`I_MPI_DAPL_UD_DESIRED_STATIC_CONNECTIONS_NUM=<num_processes>`

### Arguments

<code>&lt;num_processes&gt;</code>	Define the number of processes that establish DAPL UD static connections at the same time
<code>&gt; 0</code>	The default value is equal to <code>200</code>

### Description

Set this environment variable to control the algorithm of DAPL UD static connections establishment.

If the number of processes in an MPI job is less than or equal to `<num_processes>`, all MPI processes establish the static connections simultaneously. Otherwise, the processes are distributed into several groups. The number of processes in each group is calculated to be close to `<num_processes>`. Then static connections are established in several iterations, including intergroup connection setup.

## I\_MPI\_DAPL\_UD\_RDMA\_MIXED

Control the use of the DAPL UD/RDMA mixed communication.

### Syntax

`I_MPI_DAPL_UD_RDMA_MIXED = <arg>`

## Arguments

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Turn on the use of DAPL UD/RDMA mixed communication
<code>disable   no   off   0</code>	Turn off the use of DAPL UD/RDMA mixed communication. This is the default value

## Description

Set this environment variable to enable the DAPL UD/RDMA mixed mode for transferring data. In the DAPL UD/RDMA mixed mode, small messages are passed through the UD transport and large messages are passed through the RDMA transport. If you set the `I_MPI_DAPL_UD_RDMA_MIXED` environment variable and a certain DAPL provider attribute indicates that UD IB extension is supported, the DAPL UD/RDMA mixed mode is enabled.

The following set of `I_MPI_DAPL_UD*` environment variables also controls the DAPL UD/RDMA mixed mode:

- `I_MPI_DAPL_UD_PROVIDER`
- `I_MPI_DAPL_UD_EAGER_DYNAMIC_CONNECTION`
- `I_MPI_DAPL_UD_RNDV_DYNAMIC_CONNECTION`
- `I_MPI_DAPL_UD_DIRECT_COPY_THRESHOLD`
- `I_MPI_DAPL_UD_RECV_BUFFER_NUM`
- `I_MPI_DAPL_UD_SEND_BUFFER_NUM`
- `I_MPI_DAPL_UD_NUMBER_CREDIT_UPDATE`
- `I_MPI_DAPL_UD_ACK_SEND_POOL_SIZE`
- `I_MPI_DAPL_UD_ACK_RECV_POOL_SIZE`
- `I_MPI_DAPL_UD_RESENT_TIMEOUT`
- `I_MPI_DAPL_UD_MAX_MSG_SIZE`
- `I_MPI_DAPL_UD_SEND_BUFFER_SIZE`
- `I_MPI_DAPL_UD_REQ_EVD_SIZE`
- `I_MPI_DAPL_UD_REQUEST_QUEUE_SIZE`
- `I_MPI_DAPL_UD_MULTIPLE_EAGER_SEND`
- `I_MPI_DAPL_UD_NA_SBUF_LIMIT`
- `I_MPI_DAPL_UD_RECV_EVD_SIZE`
- `I_MPI_DAPL_UD_CONNECTION_TIMEOUT`
- `I_MPI_DAPL_UD_PORT`
- `I_MPI_DAPL_UD_CREATE_CONN_QUAL,`
- `I_MPI_DAPL_UD_FINALIZE_RETRY_COUNT`
- `I_MPI_DAPL_UD_FINALIZE_TIMEOUT`
- `I_MPI_DAPL_UD_TRANSLATION_CACHE`
- `I_MPI_DAPL_UD_TRANSLATION_CACHE_AVL_TREE`
- `I_MPI_DAPL_UD_TRANSLATION_CACHE_MAX_ENTRY_NUM`
- `I_MPI_DAPL_UD_TRANSLATION_CACHE_MAX_MEMORY_SIZE`
- `I_MPI_DAPL_UD_PKT_LOSS_OPTIMIZATION`
- `I_MPI_DAPL_UD_DFACTOR`
- `I_MPI_DAPL_UD_DESIRED_STATIC_CONNECTIONS_NUM`
- `I_MPI_DAPL_UD_CONN_EVD_SIZE`
- `I_MPI_DAPL_UD_RNDV_BUFFER_ALIGNMENT`
- `I_MPI_DAPL_UD_RNDV_COPY_ALIGNMENT_THRESHOLD`

The following set of environment variables is specific for DAPL UD/RDMA mixed mode:

- `I_MPI_DAPL_UD_MAX_RDMA_SIZE`
- `I_MPI_DAPL_UD_MAX_RDMA_DTOS`

### `I_MPI_DAPL_UD_MAX_RDMA_SIZE`

Control the maximum message size that can be sent though the RDMA for DAPL UD/RDMA mixed mode.

**Syntax**

`I_MPI_DAPL_UD_MAX_RDMA_SIZE =<nbytes>`

**Arguments**

<code>&lt;nbytes&gt;</code>	Define the maximum message size that can be sent through RDMA without fragmentation
<code>&gt; 0</code>	The default <code>&lt;nbytes&gt;</code> value is 4 MB

**Description**

Set this environment variable to define the maximum message size that can be sent though RDMA protocol for the DAPL UD/RDMA mixed mode. If the message size is greater than this value, this message is divided into several fragments and is sent by several RDMA operations.

### `I_MPI_DAPL_UD_MAX_RDMA_DTOS`

Control the maximum number of uncompleted RDMA operations per connection for the DAPL UD/RDMA mixed mode.

**Syntax**

`I_MPI_DAPL_UD_MAX_RDMA_DTOS= <arg>`

**Arguments**

<code>&lt;arg&gt;</code>	Define the maximum number of RDMA operations per connection
<code>&gt; 0</code>	The default <code>&lt;arg&gt;</code> value is 8

**Description**

Set this environment variable to define the maximum number of RDMA operations per connection for the DAPL UD/RDMA mixed mode.

## 3.3.5 TCP-capable Network Fabrics Control

### `I_MPI_TCP_NETMASK`

#### `(I_MPI_NETMASK)`

Choose the network interface for MPI communication over TCP-capable network fabrics.

**Syntax**

`I_MPI_TCP_NETMASK= <arg>`

## Arguments

<code>&lt;arg&gt;</code>	Define the network interface (string parameter)
<code>&lt;interface_mnemonic&gt;</code>	Mnemonic of the network interface: <code>ib</code> or <code>eth</code>
<code>ib</code>	Select IPoIB*
<code>eth</code>	Select Ethernet. This is the default value
<code>&lt;interface_name&gt;</code>	Name of the network interface Usually the UNIX* driver name followed by the unit number
<code>&lt;network_address&gt;&gt;</code>	Network address. The trailing zero bits imply netmask
<code>&lt;network_address/ netmask&gt;</code>	Network address. The <code>&lt;netmask&gt;</code> value specifies the netmask length
<code>&lt;list of interfaces&gt;</code>	A colon separated list of network addresses and interface names

## Description

Set this environment variable to choose the network interface for MPI communication over TCP-capable network fabrics. If you specify a list of interfaces, the first available interface on the node will be used for communication.

## Examples

- Use the following setting to select the IP over InfiniBand\* (IPoIB) fabric:  
`I_MPI_TCP_NETMASK=ib`
- Use the following setting to select the specified network interface for socket communications:  
`I_MPI_TCP_NETMASK=ib0`
- Use the following setting to select the specified network for socket communications. This setting implies the `255.255.0.0` netmask:  
`I_MPI_TCP_NETMASK=192.169.0.0`
- Use the following setting to select the specified network for socket communications with netmask set explicitly:  
`I_MPI_TCP_NETMASK=192.169.0.0/24`
- Use the following setting to select the specified network interfaces for socket communications:  
`I_MPI_TCP_NETMASK=192.169.0.5/24:ib0:192.169.0.0`

## I\_MPI\_TCP\_BUFFER\_SIZE

Change the size of the TCP socket buffers.

## Syntax

`I_MPI_TCP_BUFFER_SIZE=<nbytes>`

## Arguments

<code>&lt;nbytes&gt;</code>	Define the size of the TCP socket buffers
<code>&gt; 0</code>	The default <code>&lt;nbytes&gt;</code> value is equal to default value of the TCP socket buffer size on your Linux system.

## Description

Set this environment variable to manually define the size of the TCP socket buffers. The TCP socket buffer size is restricted by the existing TCP settings on your Linux system.

Use the `I_MPI_TCP_BUFFER_SIZE` environment variable for tuning your application's performance for a given number of processes.

**NOTE:** TCP socket buffers of a large size can require more memory for an application with large number of processes. Alternatively, TCP socket buffers of a small size can considerably decrease the bandwidth of each socket connection especially for 10 Gigabit Ethernet and IPoIB (see [I\\_MPI\\_TCP\\_NETMASK](#) for details).

### I\_MPI\_TCP\_POLLING\_MODE

Set this environment variable to define a polling mode.

**Syntax**

`I_MPI_TCP_POLLING_MODE=<mode>`

**Arguments**

<code>&lt;mode&gt;</code>	Specify the polling mode
<code>poll</code>	The polling mode based on the <code>poll()</code> function. This is <i>the</i> default value
<code>epoll[:edge]</code>	The polling mode based on the <code>epoll()</code> function as an edge-triggered interface
<code>epoll:level</code>	The polling mode based on the <code>epoll()</code> function as a level-triggered interface

Set this environment variable to select the polling mode for the `tcp` fabric.

Use the `I_MPI_TCP_POLLING_MODE` environment variable for tuning application performance. You can choose the best polling mode on an experimental basis. The best mode depends on the specific application and on the number of processes. The `epoll` polling mode is a preferable mode in the following situations:

- for large number of processes
- for APP client-server type
- for `MPI_ANY_SOURCE` tag matching

## 3.3.6 TMI-capable Network Fabrics Control

### I\_MPI\_TMI\_LIBRARY

Select the TMI library to be used.

**Syntax**

`I_MPI_TMI_LIBRARY=<library>`

**Arguments**

<code>&lt;library&gt;</code>	Specify a TMI library to be used instead of the default <code>libtmi.so</code>
------------------------------	--

**Description**

Set this environment variable to select a specific TMI library. Specify the full path to the TMI library if the library does not locate in the dynamic loader search path.

### I\_MPI\_TMI\_PROVIDER

Define the name of the TMI provider to load.

**Syntax**

`I_MPI_TMI_PROVIDER=<name>`

**Arguments**

<code>&lt;name&gt;</code>	name of the TMI provider to load
---------------------------	----------------------------------

**Description**

Set this environment variable to define the name of the TMI provider to load. The name must also be defined in the tmi.conf configuration file.

### I\_MPI\_TMI\_PROBE\_INTERVAL

Define the frequency that the TMI module probes the internal control messages.

**Syntax**

`I_MPI_TMI_PROBE_INTERVAL=<value>`

**Arguments**

<code>&lt;value&gt;</code>	Define the frequency that the TMI module probes the internal control messages
<code>integer &gt; 0</code>	Exact value for the option

**Description**

Set this environment variable to define how often the TMI module should probe for incoming internal control messages. A larger value means less frequent probes. The value `1` means that a probe happens each time the TMI module is polled for progress. The default setting is `20`.

Reducing the probe frequency helps improve the performance when there are a large number of unexpected messages. The trade-off is longer response time for the internal control messages. In MPI 4.0, the internal control messages only affect the MPI functions for one-sided operations (RMA).

## 3.3.7 OFA\*-capable Network Fabrics Control

### I\_MPI\_OFA\_NUM\_ADAPTERS

Set the number of connection adapters.

**Syntax**

`I_MPI_OFA_NUM_ADAPTERS=<arg>`

**Arguments**

<code>&lt;arg&gt;</code>	Define the maximum number of connection adapters used
<code>&gt;0</code>	Use the specified number of adapters. The default value is <code>1</code>

**Description**

Set the number of the used adapters. If the number is greater than the available number of adapters, all the available adaptors are used.

I\_MPI\_OFA\_ADAPTER\_NAME

Set the name of adapter that is used.

Syntax

I\_MPI\_OFA\_ADAPTER\_NAME=<arg>

Arguments

<arg>	Define the name of adapter
Name	Use the specified adapter. By default, any adapter can be used

Description

Set the name of adaptor to be used. If the adapter with specified name does not exist, the library returns error. This has effect only if I\_MPI\_OFA\_NUM\_ADAPTERS=1.

I\_MPI\_OFA\_NUM\_PORTS

Set the number of used ports on each adapter.

Syntax

I\_MPI\_OFA\_NUM\_PORTS= <arg>

Arguments

<arg>	Define the number of ports that are used on each adapter
> 0	Use the specified number of ports. The default value is 1

Description

Set the number of used ports on each adaptor. If the number is greater than the available number of ports, all the available ports are used.

I\_MPI\_OFA\_NUM\_RDMA\_CONNECTIONS

Set the maximum number of connections that use the rdma exchange protocol.

Syntax

I\_MPI\_OFA\_NUM\_RDMA\_CONNECTIONS= <num\_conn>

Arguments

<num_conn>	Define the maximum number of connections that use the rdma exchange protocol
>= 0	Create the specified number of connections that use the rdma exchange protocol. The rest processes use the send/ receive exchange protocol
-1	Create log <sub>2</sub> (number of processes) rdma connections
>= number of processes	Create rdma connections for all processes. This is the default value

Description

There are two exchange protocols between two processes: send/receive and rdma. This environment variable specifies the maximum amount of connections that use rdma protocol.

RDMA protocol is faster but requires more resources. For a large application, you can limit the number of connections that use the `rdma` protocol so that only processes that actively exchange data use the `rdma` protocol.

## I\_MPI\_OFA\_SWITCHING\_TO\_RDMA

Set the number of messages that a process should receive before switching this connection to RDMA exchange protocol.

### Syntax

`I_MPI_OFA_SWITCHING_TO_RDMA=<number>`

### Arguments

<code>&lt;number&gt;</code>	Define the number of messages that the process receives before switching to use the <code>rdma</code> protocol
<code>&gt;= 0</code>	If this process receives <code>&lt;number&gt;</code> of messages, start using the <code>rdma</code> protocol

### Description

Count the number of messages received from the specific process. The connection achieved the specified number tries to switch to `rdma` protocol for exchanging with that process. The connection will not switch to `rdma` protocol if the maximum number of connections that use the `rdma` exchange protocol defined in `I_MPI_OFA_NUM_RDMA_CONNECTIONS` has been reached.

## I\_MPI\_OFA\_RAIL\_SCHEDULER

Set the method of choosing rails for short messages.

### Syntax

`I_MPI_OFA_RAIL_SCHEDULER=<arg>`

### Arguments

<code>&lt;arg&gt;</code>	Mode selector
<code>ROUND_ROBIN</code>	Next time use next rail
<code>FIRST_RAIL</code>	Always use the first rail for short messages
<code>PROCESS_BIND</code>	Always use the rail specific for process

### Description

Set the method of choosing rails for short messages. The algorithms are selected according to the following scheme:

- In the `ROUND_ROBIN` mode, the first message is sent using the first rail; the next message is sent using the second rail, and so on.
- In the `FIRST_RAIL` mode, the first rail is always used for short messages.
- In the `PROCESS_BIND` mode, the process with the smallest rank uses the first rail, and the next uses the second rail.

## I\_MPI\_OFA\_TRANSLATION\_CACHE

Turn on/off the memory registration cache.

**Syntax**

`I_MPI_OFA_TRANSLATION_CACHE=<arg>`

**Arguments**

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Turn on the memory registration cache. This is the default
<code>disable   no   off   0</code>	Turn off the memory registration cache

**Description**

Set this environment variable to turn on/off the memory registration cache.

The cache substantially increases performance, but may lead to correctness issues in certain situations. See product *Release Notes* for further details.

**I\_MPI\_OFA\_TRANSLATION\_CACHE\_AVL\_TREE**

Enable/disable the AVL tree\* based implementation of the RDMA translation cache.

**Syntax**

`I_MPI_OFA_TRANSLATION_CACHE_AVL_TREE= <arg>`

**Arguments**

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Turn on the AVL tree based RDMA translation cache
<code>disable   no   off   0</code>	Turn off the AVL tree based RDMA translation cache. This is the default value

**Description**

Set this environment variable to enable the AVL tree based implementation of RDMA translation cache in the OFA path. When the search in RDMA translation cache handles over 10,000 elements, the AVL tree based RDMA translation cache is faster than the default implementation.

**I\_MPI\_OFA\_USE\_XRC**

Control the use of extensible reliable connection (XRC) capability.

**Syntax**

`I_MPI_OFA_USE_XRC=<arg>`

**Arguments**

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Turn on the use of XRC.
<code>disable   no   off   0</code>	Turn off the use of XRC. This is the default

**Description**

Set this environment variable to control the use of XRC when you are using a large cluster with several thousands of nodes.

## I\_MPI\_OFA\_DYNAMIC\_QPS

Control the library to create queue pairs (QPs) dynamically.

### Syntax

`I_MPI_OFA_DYNAMIC_QPS=<arg>`

### Arguments

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Create QPs dynamically. This is the default value if the number of processes is larger than or equal <code>2,000</code>
<code>disable   no   off   0</code>	Create all QPs during the init stage. This is the default value if the number of processes is less than <code>2,000</code>

### Description

Set this environment variable to turn on dynamic creation of QPs.

## I\_MPI\_OFA\_PACKET\_SIZE

Set the size of the packet used for sending.

### Syntax

`I_MPI_OFA_PACKET_SIZE=<arg>`

### Arguments

<code>&lt;arg&gt;</code>	Define the size of packet in bytes
<code>&gt; 0</code>	Use the specified packet size. The default value is <code>8192</code>

### Description

Set the packet size in bytes. If the number is negative, the size is set to `8`.

## I\_MPI\_OFA\_LIBRARY

Set the name of the used OFA library.

### Syntax

`I_MPI_OFA_LIBRARY=<arg>`

### Arguments

<code>&lt;arg&gt;</code>	Define the name of the OFA library
<code>Name</code>	Use the specified library. By default, the name is <code>libibverbs.so</code>

### Description

Set the name of the InfiniBand\* (IB\*) library. If the library with the specified name does not exist, an error is returned.

## I\_MPI\_OFA\_NONSWITCH\_CONF

Define the nonstandard template for port connections.

### Syntax

`I_MPI_OFA_NONSWITCH_CONF= <arg>`

## Arguments

<code>&lt;arg&gt;</code>	Define the template for port connections
<b>Name</b>	Use the specified template

## Description

The nodes in clusters are normally connected the way so that port<sub>i</sub> of a node can access port<sub>i</sub> of all other nodes. Use this environment variable if ports are connected in a nonstandard way. The following example is the template format:

```
host1@port11#port12#...#host2@port21#port22....
```

Port<sub>i</sub><sup>j</sup> defines the port used to send from host<sub>i</sub> to host<sub>j</sub>

For example:

```
node1@1#1#2#node2@2#1#1#node3@1#2#1#
```

This sample specifies the following configuration:

- Port1 of node1 connected to port2 of node2
- Port2 of node1 connected to port1 of node3
- Port1 of node2 connected to port2 of node3
- Port2 of node2 connected to port1 of node2
- Port1 of node3 connected to port2 of node1
- Port2 of node3 connected to port1 of node2

Port1 is always used to communicate with itself (loopback).

## 3.3.8 Failover Support in the OFA\* Device

The Intel® MPI Library recognizes the following events to detect hardware issues:

- `IBV_EVENT_QP_FATAL` Error occurred on a QP and it transitioned to error state
- `IBV_EVENT_QP_REQ_ERR` Invalid request local work queue error
- `IBV_EVENT_QP_ACCESS_ERR` Local access violation error
- `IBV_EVENT_PATH_MIG_ERR` A connection failed to migrate to the alternate path
- `IBV_EVENT_CQ_ERR` CQ is in error (CQ overrun)
- `IBV_EVENT_SRQ_ERR` Error occurred on an SRQ
- `IBV_EVENT_PORT_ERR` Link became unavailable on a port
- `IBV_EVENT_DEVICE_FATAL` CA is in `FATAL` state

Intel® MPI Library stops using port or whole adapter for communications if one of these issues is detected. The communications will be continued through the available port or adapter if application is running in the multi-rail mode. The application will be aborted if no healthy ports/adapters are available.

Intel® MPI Library also recognizes the following event

- `IBV_EVENT_PORT_ACTIVE` Link became active on a port

The event indicates that the port can be used again and is enabled for communications.

## 3.4 Dynamic Process Support

The Intel® MPI Library provides support for the MPI-2 process model what allows creation and cooperative termination of processes after an MPI application has started. It provides

- a mechanism to establish communication between the newly created processes and the existing MPI application
- a process attachment mechanism to establish communication between two existing MPI applications even when one of them does not spawn the other

The existing MPD ring (see [mpdboot](#) for details) is used for the placement of the spawned processes in the round robin fashion. The first spawned process is placed after the last process of the parent group. A specific network fabric combination is selected using the usual fabrics selection algorithm (see [I\\_MPI\\_FABRICS](#) and [I\\_MPI\\_FABRICS\\_LIST](#) for details).

For example, to run a dynamic application, use the following commands:

```
$ mpdboot -n 4 -r ssh

$ mpiexec -n 1 -gwdir <path_to_executable> -genv I_MPI_FABRICS shm:tcp
<spawn_app>
```

In the example, the `spawn_app` spawns 4 dynamic processes. If the `mpd.hosts` contains the following information,

```
host1
host2
host3
host4
```

the original spawning process is placed on `host1`, while the dynamic processes is distributed as follows: 1 – on `host2`, 2 – on `host3`, 3 – on `host4`, and 4 – again on `host1`.

To run a client-server application, use the following commands on the server host:

```
$ mpdboot -n 1

$ mpiexec -n 1 -genv I_MPI_FABRICS shm:dapl <server_app> > <port_name>
```

and use the following commands on the intended client hosts:

```
$ mpdboot -n 1

$ mpiexec -n 1 -genv I_MPI_FABRICS shm:dapl <client_app> < <port_name>
```

To run a simple MPI\_COMM\_JOIN based application, use the following commands on the intended host:

```
$ mpdboot -n 1 -r ssh

$ mpiexec -n 1 -genv I_MPI_FABRICS shm:ofa <join_server_app> <
<port_number>

$ mpiexec -n 1 -genv I_MPI_FABRICS shm:ofa <join_client_app> <
<port_number>
```

## 3.5 Collective Operation Control

Each collective operation in the Intel® MPI Library supports a number of communication algorithms. In addition to highly optimized default settings, the library provides two ways to control the algorithm selection explicitly: the novel `I_MPI_ADJUST` environment variable family and the deprecated `I_MPI_MSG` environment variable family. They are described in the following sections.

These environment variables are available for both Intel® and non-Intel microprocessors, but they may perform additional optimizations for Intel microprocessors than they perform for non-Intel microprocessors.

### 3.5.1 I\_MPI\_ADJUST Family

#### I\_MPI\_ADJUST\_<opname>

Control collective operation algorithm selection.

##### Syntax

`I_MPI_ADJUST_<opname>=<algid>[:<conditions>][;<algid>:<conditions>[...]]`

##### Arguments

<code>&lt;algid&gt;</code>	Algorithm identifier
<code>&gt;= 0</code>	The default value of zero selects the reasonable settings
<code>&lt;conditions&gt;</code>	A comma separated list of conditions. An empty list selects all message sizes and process combinations
<code>&lt;l&gt;</code>	Messages of size <code>&lt;l&gt;</code>
<code>&lt;l&gt;-&lt;m&gt;</code>	Messages of size from <code>&lt;l&gt;</code> to <code>&lt;m&gt;</code> , inclusive
<code>&lt;l&gt;@&lt;p&gt;</code>	Messages of size <code>&lt;l&gt;</code> and number of processes <code>&lt;p&gt;</code>
<code>&lt;l&gt;-&lt;m&gt;@&lt;p&gt;-&lt;q&gt;</code>	Messages of size from <code>&lt;l&gt;</code> to <code>&lt;m&gt;</code> and number of processes from <code>&lt;p&gt;</code> to <code>&lt;q&gt;</code> , inclusive

##### Description

Set this environment variable to select the desired algorithm(s) for the collective operation `<opname>` under particular conditions. Each collective operation has its own environment variable and algorithms. See below.

Table 3.5-1 Environment Variables, Collective Operations, and Algorithms

Environment Variable	Collective Operation	Algorithms
<code>I_MPI_ADJUST_ALLGATHER</code>	<code>MPI_Allgather</code>	<ol style="list-style-type: none"><li>1. Recursive doubling algorithm</li><li>2. Bruck's algorithm</li><li>3. Ring algorithm</li><li>4. Topology aware Gather + Bcast algorithm</li></ol>

I_MPI_ADJUST_ALLGATHERV	MPI_Allgatherv	<ol style="list-style-type: none"> <li>1. Recursive doubling algorithm</li> <li>2. Bruck's algorithm</li> <li>3. Ring algorithm</li> <li>4. Topology aware Gatherv + Bcast algorithm</li> </ol>
I_MPI_ADJUST_ALLREDUCE	MPI_Allreduce	<ol style="list-style-type: none"> <li>1. Recursive doubling algorithm</li> <li>2. Rabenseifner's algorithm</li> <li>3. Reduce + Bcast algorithm</li> <li>4. Topology aware Reduce + Bcast algorithm</li> <li>5. Binomial gather + scatter algorithm</li> <li>6. Topology aware binominal gather + scatter algorithm</li> <li>7. Shumilin's ring algorithm</li> <li>8. Ring algorithm</li> </ol>
I_MPI_ADJUST_ALLTOALL	MPI_Alltoall	<ol style="list-style-type: none"> <li>1. Bruck's algorithm</li> <li>2. Isend/Irecv + waitall algorithm</li> <li>3. Pair wise exchange algorithm</li> <li>4. Plum's algorithm</li> </ol>
I_MPI_ADJUST_ALLTOALLV	MPI_Alltoallv	<ol style="list-style-type: none"> <li>1. Isend/Irecv + waitall algorithm</li> <li>2. Plum's algorithm</li> </ol>
I_MPI_ADJUST_ALLTOALLW	MPI_Alltoallw	<ol style="list-style-type: none"> <li>1. Isend/Irecv + waitall algorithm</li> </ol>
I_MPI_ADJUST_BARRIER	MPI_Barrier	<ol style="list-style-type: none"> <li>1. Dissemination algorithm</li> <li>2. Recursive doubling algorithm</li> <li>3. Topology aware dissemination algorithm</li> <li>4. Topology aware recursive doubling algorithm</li> <li>5. Binominal gather + scatter algorithm</li> <li>6. Topology aware binominal gather + scatter algorithm</li> </ol>
I_MPI_ADJUST_BCAST	MPI_Bcast	<ol style="list-style-type: none"> <li>1. Binomial algorithm</li> <li>2. Recursive doubling algorithm</li> <li>3. Ring algorithm</li> <li>4. Topology aware binomial algorithm</li> <li>5. Topology aware recursive doubling algorithm</li> <li>6. Topology aware ring algorithm</li> <li>7. Shumilin's bcast algorithm</li> </ol>
I_MPI_ADJUST_EXSCAN	MPI_Exscan	<ol style="list-style-type: none"> <li>1. Partial results gathering algorithm</li> <li>2. Partial results gathering regarding algorithm layout of processes</li> </ol>

I_MPI_ADJUST_GATHER	MPI_Gather	<ol style="list-style-type: none"> <li>1. Binomial algorithm</li> <li>2. Topology aware binomial algorithm</li> <li>3. Shumilin's algorithm</li> </ol>
I_MPI_ADJUST_GATHERV	MPI_Gatherv	<ol style="list-style-type: none"> <li>1. Linear algorithm</li> <li>2. Topology aware linear algorithm</li> </ol>
I_MPI_ADJUST_REDUCE_SCATTER	MPI_Reduce_scatter	<ol style="list-style-type: none"> <li>1. Recursive having algorithm</li> <li>2. Pair wise exchange algorithm</li> <li>3. Recursive doubling algorithm</li> <li>4. Reduce + Scatterv algorithm</li> <li>5. Topology aware Reduce + Scatterv algorithm</li> </ol>
I_MPI_ADJUST_REDUCE	MPI_Reduce	<ol style="list-style-type: none"> <li>1. Shumilin's algorithm</li> <li>2. Binomial algorithm</li> <li>3. Topology aware Shumilin's algorithm</li> <li>4. Topology aware binomial algorithm</li> <li>5. Rabenseifner's algorithm</li> <li>6. Topology aware Rabenseifner's algorithm</li> </ol>
I_MPI_ADJUST_SCAN	MPI_Scan	<ol style="list-style-type: none"> <li>1. Partial results gathering algorithm</li> <li>2. Topology aware partial results gathering algorithm</li> </ol>
I_MPI_ADJUST_SCATTER	MPI_Scatter	<ol style="list-style-type: none"> <li>1. Binomial algorithm</li> <li>2. Topology aware binomial algorithm</li> <li>3. Shumilin's algorithm</li> </ol>
I_MPI_ADJUST_SCATTERV	MPI_Scatterv	<ol style="list-style-type: none"> <li>1. Linear algorithm</li> <li>2. Topology aware linear algorithm</li> </ol>

The message size calculation rules for the collective operations are described in the table below. Here, "n/a" means that the corresponding interval  $\langle l \rangle - \langle m \rangle$  should be omitted.

**Table 3.5-2 Message Collective Functions**

Collective Function	Message Size Formula
MPI_Allgather	$\text{recv\_count} * \text{recv\_type\_size}$
MPI_Allgatherv	$\text{total\_recv\_count} * \text{recv\_type\_size}$
MPI_Allreduce	$\text{count} * \text{type\_size}$
MPI_Alltoall	$\text{send\_count} * \text{send\_type\_size}$
MPI_Alltoallv	n/a
MPI_Alltoallw	n/a

<code>MPI_Barrier</code>	n/a
<code>MPI_Bcast</code>	<code>count*type_size</code>
<code>MPI_Exscan</code>	<code>count*type_size</code>
<code>MPI_Gather</code>	<code>recv_count*recv_type_size</code> if <code>MPI_IN_PLACE</code> is used, otherwise <code>send_count*send_type_size</code>
<code>MPI_Gatherv</code>	n/a
<code>MPI_Reduce_scatter</code>	<code>total_recv_count*type_size</code>
<code>MPI_Reduce</code>	<code>count*type_size</code>
<code>MPI_Scan</code>	<code>count*type_size</code>
<code>MPI_Scatter</code>	<code>send_count*send_type_size</code> if <code>MPI_IN_PLACE</code> is used, otherwise <code>recv_count*recv_type_size</code>
<code>MPI_Scatterv</code>	n/a

### Examples

1. Use the following settings to select the second algorithm for `MPI_Reduce` operation:  
`I_MPI_ADJUST_REDUCE=2`
2. Use the following settings to define the algorithms for `MPI_Reduce_scatter` operation:  
`I_MPI_ADJUST_REDUCE_SCATTER=4:0-100,5001-10000;1:101-3200,2:3201-5000;3`

In this case, algorithm 4 will be used for the message sizes from 0 up to 100 bytes and from 5001 to 10000 bytes, algorithm 1 will be used for the message sizes from 101 up to 3200 bytes, algorithm 2 will be used for the message sizes from 3201 up to 5000 bytes, and algorithm 3 will be used for all other messages.

## 3.5.2 I\_MPI\_MSG Family

These environment variables are deprecated and supported mostly for backward compatibility. Use the `I_MPI_ADJUST` environment variable family whenever possible.

### I\_MPI\_FAST\_COLLECTIVES

Control the default library behavior during selection of the most appropriate collective algorithm.

#### Syntax

`I_MPI_FAST_COLLECTIVES=<arg>`

#### Arguments

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Fast collective algorithms are used. This is the default value
<code>disable   no   off   0</code>	Slower and safer collective algorithms are used

#### Description

The Intel® MPI Library uses advanced collective algorithms designed for better application performance by default. The implementation makes the following assumptions:

- It is safe to utilize the flexibility of the MPI standard regarding the order of execution of the collective operations to take advantage of the process layout and other opportunities.

- There is enough memory available for allocating additional internal buffers.

Set the `I_MPI_FAST_COLLECTIVES` environment variable to `disable` if you need to obtain results that do not depend on the physical process layout or other factors.

**NOTE:** Some optimizations controlled by this environment variable are of an experimental nature. In case of failure, turn off the collective optimizations and repeat the run.

### I\_MPI\_BCAST\_NUM\_PROCS

Control `MPI_Bcast` algorithm thresholds.

#### Syntax

`I_MPI_BCAST_NUM_PROCS=<nproc>`

#### Arguments

<code>&lt;nproc&gt;</code>	Define the number of processes threshold for choosing the <code>MPI_Bcast</code> algorithm
<code>&gt; 0</code>	The default value is 8

### I\_MPI\_BCAST\_MSG

Control `MPI_Bcast` algorithm thresholds.

#### Syntax

`I_MPI_BCAST_MSG=<nbytes1,nbytes2>`

#### Arguments

<code>&lt;nbytes1,nbytes2&gt;</code>	Define the message size threshold range (in bytes) for choosing the <code>MPI_Bcast</code> algorithm
<code>&gt; 0</code> <code>nbytes2 &gt;= nbytes1</code>	The default pair of values is 12288,524288

#### Description

Set these environment variables to control the selection of the three possible `MPI_Bcast` algorithms according to the following scheme (See Table 3.5-1 for algorithm descriptions):

1. The first algorithm is selected if the message size is less than `<nbytes1>`, or the number of processes in the operation is less than `<nproc>`.
2. The second algorithm is selected if the message size is greater than or equal to `<nbytes1>` and less than `<nbytes2>`, and the number of processes in the operation is a power of two.
3. If none of the above conditions is satisfied, the third algorithm is selected.

### I\_MPI\_ALLTOALL\_NUM\_PROCS

Control `MPI_Alltoall` algorithm thresholds.

#### Syntax

`I_MPI_ALLTOALL_NUM_PROCS=<nproc>`

#### Arguments

<code>&lt;nproc&gt;</code>	Define the number of processes threshold for choosing the <code>MPI_Alltoall</code> algorithm
----------------------------	---

> 0	The default value is 8
-----	------------------------

## I\_MPI\_ALLTOALL\_MSG

Control `MPI_Alltoall` algorithm thresholds.

### Syntax

`I_MPI_ALLTOALL_MSG=<nbytes1,nbytes2>`

### Arguments

<code>&lt;nbytes1,nbytes2&gt;</code>	Defines the message size threshold range (in bytes) for choosing the <code>MPI_Alltoall</code> algorithm
<code>&gt; 0</code> <code>nbytes2 &gt;= nbytes1</code>	The default pair of values is 256,32768

### Description

Set these environment variables to control the selection of the three possible `MPI_Alltoall` algorithms according to the following scheme (See Table 3.5-1 for algorithm descriptions):

1. The first algorithm is selected if the message size is greater than or equal to `<nbytes1>`, and the number of processes in the operation is not less than `<nproc>`.
2. The second algorithm is selected if the message size is greater than `<nbytes1>` and less than or equal to `<nbytes2>`, or if the message size is less than `<nbytes2>` and the number of processes in the operation is less than `<nproc>`.
3. If none of the above conditions is satisfied, the third algorithm is selected.

## I\_MPI\_ALLGATHER\_MSG

Control `MPI_Allgather` algorithm thresholds.

### Syntax

`I_MPI_ALLGATHER_MSG=<nbytes1,nbytes2>`

### Arguments

<code>&lt;nbytes1,nbytes2&gt;</code>	Define the message size threshold range (in bytes) for choosing the <code>MPI_Allgather</code> algorithm
<code>&gt; 0</code> <code>nbytes2 &gt;= nbytes1</code>	The default pair of values is 81920,524288

### Description

Set this variable to control the selection of the three possible `MPI_Allgather` algorithms according to the following scheme (See Table 3.5-1 for algorithm descriptions):

1. The first algorithm is selected if the message size is less than `<nbytes2>` and the number of processes in the operation is a power of two.
2. The second algorithm is selected if the message size is less than `<nbytes1>` and number of processes in the operation is not a power of two.
3. If none of the above conditions is satisfied, the third algorithm is selected.

## I\_MPI\_ALLREDUCE\_MSG

Control `MPI_Allreduce` algorithm thresholds.

## Syntax

`I_MPI_ALLREDUCE_MSG=<nbytes>`

## Arguments

<code>&lt;nbytes&gt;</code>	Define the message size threshold (in bytes) for choosing the <code>MPI_Allreduce</code> algorithm
<code>&gt; 0</code>	The default value is <code>2048</code>

## Description

Set this environment variable to control the selection of the two possible `MPI_Allreduce` algorithms according to the following scheme (See Table 3.5-1 for algorithm descriptions):

1. The first algorithm is selected if the message size is less than or equal `<nbytes>`, or the reduction operation is user-defined, or the count argument is less than the nearest power of two less than or equal to the number of processes.
2. If the above condition is not satisfied, the second algorithm is selected.

## `I_MPI_REDS CAT_MSG`

Control the `MPI_Reduce_scatter` algorithm thresholds.

## Syntax

`I_MPI_REDS CAT_MSG=<nbytes1,nbytes2>`

## Arguments

<code>&lt;nbytes&gt;</code>	Define the message size threshold range (in bytes) for choosing the <code>MPI_Reduce_scatter</code> algorithm
<code>&gt; 0</code>	The default pair of values is <code>512,524288</code>

## Description

Set this environment variable to control the selection of the three possible `MPI_Reduce_scatter` algorithms according to the following scheme (See Table 3.5-1 for algorithm descriptions):

1. The first algorithm is selected if the reduction operation is commutative and the message size is less than `<nbytes2>`.
2. The second algorithm is selected if the reduction operation is commutative and the message size is greater than or equal to `<nbytes2>`, or if the reduction operation is not commutative and the message size is greater than or equal to `<nbytes1>`.
3. If none of the above conditions is satisfied, the third algorithm is selected.

## `I_MPI_SCATTER_MSG`

Control `MPI_Scatter` algorithm thresholds.

## Syntax

`I_MPI_SCATTER_MSG=<nbytes>`

## Arguments

<code>&lt;nbytes&gt;</code>	Define the buffer size threshold range (in bytes) for choosing the <code>MPI_Scatter</code> algorithm
<code>&gt; 0</code>	The default value is <code>2048</code>

Description

Set this environment variable to control the selection of the two possible `MPI_Scatter` algorithms according to the following scheme (See Table 3.5-1 for algorithm descriptions):

- 1. The first algorithm is selected on the intercommunicators if the message size is greater than `<nbytes>`.
- 2. If the above condition is not satisfied, the second algorithm is selected.

I\_MPI\_GATHER\_MSG

Control `MPI_Gather` algorithm thresholds.

Syntax

`I_MPI_GATHER_MSG=<nbytes>`

Arguments

<code>&lt;nbytes&gt;</code>	Define the buffer size threshold range (in bytes) for choosing the <code>MPI_Gather</code> algorithm
<code>&gt; 0</code>	The default value is <code>2048</code>

Description

Set this environment variable to control the selection of the two possible `MPI_Gather` algorithms according to the following scheme (See Table 3.5-1 for algorithm descriptions):

- 1. The first algorithm is selected on the intercommunicators if the message size is greater than `<nbytes>`.
- 2. If the above condition is not satisfied, the second algorithm is selected.

3.6 Extended File System Support

The Intel® MPI Library provides loadable shared modules to provide native support for the following file systems:

- Panasas\* ActiveScale\* File System (PanFS)
- Parallel Virtual File System, Version 2 (Pvfs2)
- Lustre\* File System

Set the `I_MPI_EXTRA_FILESYSTEM` environment variable to `on` to enable parallel file system support. Set the `I_MPI_EXTRA_FILESYSTEM_LIST` environment variable to request native support for the specific file system. For example, to request native support for Panasas\* ActiveScale\* File System, do the following:

```
$ mpiexec -env I_MPI_EXTRA_FILESYSTEM on \  
-env I_MPI_EXTRA_FILESYSTEM_LIST panfs -n 2 ./test
```

3.6.1 Environment Variables

I\_MPI\_EXTRA\_FILESYSTEM

Turn on/off native parallel file systems support.

## Syntax

`I_MPI_EXTRA_FILESYSTEM=<arg>`

## Arguments

<code>&lt;arg&gt;</code>	Binary indicator
<code>enable   yes   on   1</code>	Turn on native support for the parallel file systems
<code>disable   no   off   0</code>	Turn off native support for the parallel file systems. This is the default value

## Description

Set this environment variable to enable parallel file system support. The `I_MPI_EXTRA_FILESYSTEM_LIST` environment variable must be set to request native support for the specific file system.

## I\_MPI\_EXTRA\_FILESYSTEM\_LIST

Select specific file systems support.

## Syntax

`I_MPI_EXTRA_FILESYSTEM_LIST=<fs>[, <fs>, ... , <fs>]`

## Arguments

<code>&lt;fs&gt;</code>	Define a target file system
<code>panfs</code>	Panasas* ActiveScale* File System
<code>pvfs2</code>	Parallel Virtual File System, Version 2
<code>lustre</code>	Lustre* File System

## Description

Set this environment variable to request support for the specific parallel file system. This environment variable is handled only if the `I_MPI_EXTRA_FYLESYSTEM` is enabled. The Intel® MPI Library will try to load shared modules to support the file systems specified by `I_MPI_EXTRA_FILESYSTEM_LIST`.

# 3.7 Compatibility Control

The Intel® MPI Library 4.0 implements the MPI-2.1 standard. The following MPI routines are changed:

- `MPI_Cart_create`
- `MPI_Cart_map`
- `MPI_Cart_sub`
- `MPI_Graph_create`

If your application depends on the strict pre-MPI-2.1 behavior, set the environment variable `I_MPI_COMPATIBILITY` to 3.

## I\_MPI\_COMPATIBILITY

Select the runtime compatibility mode.

**Syntax**`I_MPI_COMPATIBILITY=<value>`**Arguments**

<code>&lt;value&gt;</code>	Define compatibility mode
<code>3</code>	Enable the Intel® MPI Library 3.x compatibility mode
<code>4</code>	Disable the Intel® MPI Library 3.x compatibility mode and enable the Intel® MPI Library 4.0 compatible mode. This is the default value

**Description**

Set this environment variable to choose the Intel® MPI runtime compatible mode.

## 3.8 Miscellaneous

**I\_MPI\_TIMER\_KIND**

Select the timer used by the `MPI_Wtime` and `MPI_Wtick` calls.

**Syntax**`I_MPI_TIMER_KIND=<timernam>`**Arguments**

<code>&lt;timernam&gt;</code>	Define the timer type
<code>gettimeofday</code>	If this setting is chosen, the <code>MPI_Wtime</code> and <code>MPI_Wtick</code> functions will work through the function <code>gettimeofday(2)</code> . This is the default value
<code>rdtsc</code>	If this setting is chosen, the <code>MPI_Wtime</code> and <code>MPI_Wtick</code> functions will use the high resolution RDTSC timer

**Description**

Set this environment variable to select either the ordinary or RDTSC timer.

**NOTE:** The resolution of the default `gettimeofday(2)` timer may be insufficient on certain platforms.

## 4 Statistics Gathering Mode

### 4.1 Native Statistics Format

The Intel® MPI Library has a built-in statistics gathering facility that collects essential performance data without disturbing the application execution. The collected information is output onto a text file. This section describes the environment variables used to control the built-in statistics gathering facility, and provides example output files.

#### I\_MPI\_STATS

Control statistics collection. Expand values of `I_MPI_STATS` environment variable additionally to existing values.

##### Syntax

`I_MPI_STATS=[n-] m`

##### Arguments

<code>n, m</code>	Possible stats levels of the output information
<code>1</code>	Output the amount of data sent by each process
<code>2</code>	Output the number of calls and amount of transferred data
<code>3</code>	Output statistics combined according to the actual arguments
<code>4</code>	Output statistics defined by a buckets list
<code>10</code>	Output collective operation statistics for all communication contexts

##### Description

Set this environment variable to control the amount of the statistics information collected and output onto the log file. No statistics are output by default.

**NOTE:** `n, m` represent the positive integer numbers define range of output information. The statistics from level `n` to level `m` inclusive are output. Omitted `n` value assumes to be 1.

#### I\_MPI\_STATS\_SCOPE

Select the subsystem(s) to collect statistics for.

##### Syntax

`I_MPI_STATS_SCOPE=<subsystem>[:<ops>][:<subsystem>[:<ops>]][...]`

##### Arguments

<code>&lt;subsystem&gt;</code>	Define the target subsystem(s)
<code>all</code>	Collect statistics data for all operations. This is the default value
<code>coll</code>	Collect statistics data for all collective operations
<code>p2p</code>	Collect statistics data for all point-to-point operations

<code>&lt;ops&gt;</code>	Define the target operations as a comma separated list
<code>Allgather</code>	<code>MPI_Allgather</code>
<code>Allgatherv</code>	<code>MPI_Allgatherv</code>
<code>Allreduce</code>	<code>MPI_Allreduce</code>
<code>Alltoall</code>	<code>MPI_Alltoall</code>
<code>Alltoallv</code>	<code>MPI_Alltoallv</code>
<code>Alltoallw</code>	<code>MPI_Alltoallw</code>
<code>Barrier</code>	<code>MPI_Barrier</code>
<code>Bcast</code>	<code>MPI_Bcast</code>
<code>Exscan</code>	<code>MPI_Exscan</code>
<code>Gather</code>	<code>MPI_Gather</code>
<code>Gatherv</code>	<code>MPI_Gatherv</code>
<code>Reduce_scatter</code>	<code>MPI_Reduce_scatter</code>
<code>Reduce</code>	<code>MPI_Reduce</code>
<code>Scan</code>	<code>MPI_Scan</code>
<code>Scatter</code>	<code>MPI_Scatter</code>
<code>Scatterv</code>	<code>MPI_Scatterv</code>
<code>Send</code>	Standard transfers ( <code>MPI_Send</code> , <code>MPI_Isend</code> , <code>MPI_Send_init</code> )
<code>Bsend</code>	Buffered transfers ( <code>MPI_Bsend</code> , <code>MPI_Ibsend</code> , <code>MPI_Bsend_init</code> )
<code>Csend</code>	Point-to-point operations inside the collectives. This internal operation serves all collectives
<code>Rsend</code>	Ready transfers ( <code>MPI_Rsend</code> , <code>MPI_Irsend</code> , <code>MPI_Rsend_init</code> )
<code>Ssend</code>	Synchronous transfers ( <code>MPI_Ssend</code> , <code>MPI_Issend</code> , <code>MPI_Ssend_init</code> )

## Description

Set this environment variable to select the target subsystem to collect statistics for. All collective and point-to-point operations, including the point-to-point operations performed inside the collectives are covered by default.

## Examples

1. The default settings are equivalent to:  
`I_MPI_STATS_SCOPE=coll;p2p`
2. Use the following settings to collect statistics for the `MPI_Bcast`, `MPI_Reduce`, and all point-to-point operations:  
`I_MPI_STATS_SCOPE=p2p;coll:bcast,reduce`
3. Use the following settings to collect statistics for the point-to-point operations inside the collectives:  
`I_MPI_STATS_SCOPE=p2p:csend`

## I\_MPI\_STATS\_BUCKETS

Identify a list of ranges for message sizes and communicator sizes that will be used for collecting statistics.

## Syntax

```
I_MPI_STATS_BUCKETS=<msg>[@<proc>][ , <msg>[@<proc>] ]...
```

## Arguments

<i>&lt;msg&gt;</i>	Specify range of message sizes in bytes
<i>&lt;l&gt;</i>	Single value of message size
<i>&lt;l&gt;-&lt;m&gt;</i>	Range from <i>&lt;l&gt;</i> to <i>&lt;m&gt;</i>

<i>&lt;proc&gt;</i>	Specify range of processes (ranks) for collective operations
<i>&lt;p&gt;</i>	Single value of communicator size
<i>&lt;p&gt;-&lt;q&gt;</i>	Range from <i>&lt;p&gt;</i> to <i>&lt;q&gt;</i>

## Description

Set the `I_MPI_STATS_BUCKETS` environment variable to define a set of ranges for message sizes and communicator sizes.

Level 4 of the statistics provides profile information for these ranges.

If `I_MPI_STATS_BUCKETS` environment variable is not used, then level 4 statistics is not gathered.

If a range is omitted then the maximum possible range is assumed.

## Examples

To specify short messages (from 0 to 1000 bytes) and long messages (from 50000 to 100000 bytes), use the following setting:

```
-env I_MPI_STATS_BUCKETS 0-1000,50000-100000
```

To specify messages that have 16 bytes in size and circulate within four process communicators, use the following setting:

```
-env I_MPI_STATS_BUCKETS "16@4"
```

**NOTE:** When the @ symbol is present, the environment variable value must be enclosed in quotes.

## I\_MPI\_STATS\_FILE

Define the statistics output file name.

## Syntax

```
I_MPI_STATS_FILE=<name>
```

## Arguments

<i>&lt;name&gt;</i>	Define the statistics output file name
---------------------	--

## Description

Set this environment variable to define the statistics output file. The stats.txt file is created in the current directory by default.

The statistics data is blocked and ordered according to the process ranks in the `MPI_COMM_WORLD` communicator. The timing data is presented in microseconds. For example, with the following settings in effect

```
I_MPI_STATS=4
I_MPI_STATS_SCOPE=p2p;coll:allreduce
```

the statistics output for a simple program that performs only one `MPI_Allreduce` operation may look as follows:

```
Intel(R) MPI Library Version 4.0
_____ MPI Communication Statistics _____

Stats level: 4
P2P scope:< FULL >
Collectives scope:< Allreduce >

~~~~~ Process 0 of 2 on node svlmpihead01 lifetime = 414.13

Data Transfers
Src      Dst      Amount(MB)    Transfers
-----
000 --> 000      0.000000e+00  0
000 --> 001      7.629395e-06  2
=====
Totals                7.629395e-06  2

Communication Activity
Operation      Volume(MB)    Calls
-----
P2P
Csend          7.629395e-06  2
Send           0.000000e+00  0
Bsend          0.000000e+00  0
Rsend          0.000000e+00  0
Ssend          0.000000e+00  0
Collectives
Allreduce       7.629395e-06      2
=====

Communication Activity by actual args
P2P
Operation      Dst      Message size Calls
-----
Csend
1          1          4          2
Collectives
Operation      Context      Algo    Comm size    Message size Calls    Cost(%)
-----
```

```
Allreduce
1          0          1    2          4          2      44.96
=====

~~~~ Process 1 of 2 on node svlmpihead01 lifetime = 306.13

Data Transfers
Src      Dst      Amount(MB)    Transfers
-----
001 --> 000      7.629395e-06  2
001 --> 001      0.000000e+00  0
=====
Totals          7.629395e-06  2

Communication Activity
Operation      Volume(MB)    Calls
-----
P2P
Csend          7.629395e-06  2
Send           0.000000e+00  0
Bsend          0.000000e+00  0
Rsend          0.000000e+00  0
Ssend          0.000000e+00  0
Collectives
Allreduce      7.629395e-06      2
=====

Communication Activity by actual args
P2P
Operation      Dst      Message size Calls
-----
Csend
1          0          4          2
Collectives
Operation      Context      Comm size      Message size Calls  Cost(%)
-----
Allreduce
1          0          2          4          2      37.93
=====

_____ End of stats.txt file _____
```

In the example above all times are measured in microseconds. The message sizes are counted in bytes. **MB** means megabyte equal to  $2^{20}$  or 1 048 576 bytes. The process life time is calculated as a stretch of time between `MPI_Init` and `MPI_Finalize`. The **Algo** field indicates the number of algorithm used by this operation with listed arguments. The **Cost** field represents a particular collective operation execution time as a percentage of the process life time.

## 4.2 IPM Statistics Format

The Intel® MPI Library supports integrated performance monitoring (IPM) summary format as part of the built-in statistics gathering mechanism described above. You do not need to modify the source code or re-link your application to collect this information.

The `I_MPI_STATS_BUCKETS` environment variable is not applicable to the IPM format. The `I_MPI_STATS_ACCURACY` environment variable is available to control extra functionality.

The Intel® MPI Library also supports an optional `ipm` region feature. This feature requires the source code modification. The `MPI_Pcontrol` function can be used.

### 4.2.1 Region Control

Region is a named part of the source code marked by the start/end points through the standard `MPI_Pcontrol` function calls. The `MPI_Pcontrol` function isn't used for the following special permanent regions:

- Main region contains statistics information about all MPI calls from `MPI_Init` to `MPI_Finalize`. Main region gets the `"*"` name in output.
- Complementary region contains statistics information not included into any named region. The region gets the `"ipm_noregion"` name in output.

If named regions are not used, the main regions and the complementary regions are identical and the complementary region is ignored.

Each region contains its own independent statistics information about MPI functions called inside the region.

The Intel® MPI Library supports the following types of regions:

- Discontiguous (several open and close).
- Intersected.
- Covering a subset of MPI processes (part of the `MPI_COMM_WORLD` environment variable).

A region is opened by the `MPI_Pcontrol(1, name)` call and closed by the `MPI_Pcontrol(-1, name)` call where `name` is a zero terminated string with the region name.

All open regions are closed automatically inside the `MPI_Finalize` environment variable.

### `I_MPI_STATS`

Control the statistics data output format.

#### Syntax

`I_MPI_STATS=<level>`

#### Argument

<code>&lt;level&gt;</code>	Level of statistics data
<code>ipm</code>	Summary data throughout all regions
<code>ipm:terse</code>	Basic summary data

## Description

Set this environment variable to `ipm` to get the statistics output that contains region summary. Set this environment variable to `ipm:terse` argument to get the brief statistics output.

## I\_MPI\_STATS\_FILE

Define the output file name.

### Syntax

`I_MPI_STATS_FILE=<name>`

### Argument

<code>&lt;name&gt;</code>	File name for statistics data gathering
---------------------------	---

### Description

Set this environment variable to change the statistics output file name from the default name of `stats.ipm`.

## I\_MPI\_STATS\_SCOPE

Define a semicolon separated list of subsets of MPI functions for statistics gathering.

### Syntax

`I_MPI_STATS_SCOPE=<subset>[;<subset>[;...]]`

### Argument

<code>&lt;subset&gt;</code>	Target subset
<code>all2all</code>	Collect statistics data for all to all kind of collective functions
<code>all2one</code>	Collect statistics data for all to one kind of collective functions
<code>attr</code>	Collect statistics data for attribute control functions
<code>comm</code>	Collect statistics data for communicator control functions
<code>err</code>	Collect statistics data for error handling functions
<code>group</code>	Collect statistics data for group support functions
<code>init</code>	Collect statistics data for initialize/finalize functions
<code>io</code>	Collect statistics data for input/output support function
<code>one2all</code>	Collect statistics data for one to all kind of collective functions
<code>recv</code>	Collect statistics data for receive functions
<code>req</code>	Collect statistics data for request support functions
<code>rma</code>	Collect statistics data for one sided communication functions
<code>scan</code>	Collect statistics data for scan collective functions
<code>send</code>	Collect statistics data for send functions
<code>sendrecv</code>	Collect statistics data for send/receive functions
<code>serv</code>	Collect statistics data for additional service functions
<code>spawn</code>	Collect statistics data for dynamic process functions
<code>status</code>	Collect statistics data for status control function
<code>sync</code>	Collect statistics data for barrier synchronization

time	Collect statistics data for timing support functions
topo	Collect statistics data for topology support functions
type	Collect statistics data for data type support functions

### Description

Use this environment variable to define a subset or subsets of MPI functions for statistics gathering specified by the following table. A union of all subsets is used by default.

Table 4.2-1 Stats Subsets of MPI Functions

<b>all2all</b> <i>MPI_Allgather</i> <i>MPI_Allgatherv</i> <i>MPI_Allreduce</i> <i>MPI_Alltoall</i> <i>MPI_Alltoallv</i> <i>MPI_Alltoallw</i> <i>MPI_Reduce_scatter</i>	<i>MPI_File_get_errhandler</i> <i>MPI_File_set_errhandler</i> <i>MPI_Win_call_errhandler</i> <i>MPI_Win_create_errhandler</i> <i>MPI_Win_get_errhandler</i> <i>MPI_Win_set_errhandler</i>
<b>all2one</b> <i>MPI_Gather</i> <i>MPI_Gatherv</i> <i>MPI_Reduce</i>	<b>group</b> <i>MPI_Group_compare</i> <i>MPI_Group_difference</i> <i>MPI_Group_excl</i> <i>MPI_Group_free</i> <i>MPI_Group_incl</i> <i>MPI_Group_intersection</i> <i>MPI_Group_range_excl</i> <i>MPI_Group_range_incl</i> <i>MPI_Group_rank</i> <i>MPI_Group_size</i> <i>MPI_Group_translate_ranks</i> <i>MPI_Group_union</i>
<b>attr</b> <i>MPI_Comm_create_keyval</i> <i>MPI_Comm_delete_attr</i> <i>MPI_Comm_free_keyval</i> <i>MPI_Comm_get_attr</i> <i>MPI_Comm_set_attr</i> <i>MPI_Comm_get_name</i> <i>MPI_Comm_set_name</i> <i>MPI_Type_create_keyval</i> <i>MPI_Type_delete_attr</i> <i>MPI_Type_free_keyval</i> <i>MPI_Type_get_attr</i> <i>MPI_Type_get_name</i> <i>MPI_Type_set_attr</i> <i>MPI_Type_set_name</i> <i>MPI_Win_create_keyval</i> <i>MPI_Win_delete_attr</i> <i>MPI_Win_free_keyval</i> <i>MPI_Win_get_attr</i> <i>MPI_Win_get_name</i> <i>MPI_Win_set_attr</i> <i>MPI_Win_set_name</i> <i>MPI_Get_processor_name</i>	<b>init</b> <i>MPI_Init</i> <i>MPI_Init_thread</i> <i>MPI_Finalize</i>
<b>comm</b> <i>MPI_Comm_compare</i> <i>MPI_Comm_create</i> <i>MPI_Comm_dup</i> <i>MPI_Comm_free</i> <i>MPI_Comm_get_name</i> <i>MPI_Comm_group</i> <i>MPI_Comm_rank</i> <i>MPI_Comm_remote_group</i> <i>MPI_Comm_remote_size</i> <i>MPI_Comm_set_name</i> <i>MPI_Comm_size</i> <i>MPI_Comm_split</i> <i>MPI_Comm_test_inter</i> <i>MPI_Intercomm_create</i> <i>MPI_Intercomm_merge</i>	<b>io</b> <i>MPI_File_close</i> <i>MPI_File_delete</i> <i>MPI_File_get_amode</i> <i>MPI_File_get_atomicity</i> <i>MPI_File_get_byte_offset</i> <i>MPI_File_get_group</i> <i>MPI_File_get_info</i> <i>MPI_File_get_position</i> <i>MPI_File_get_position_shared</i> <i>MPI_File_get_size</i> <i>MPI_File_get_type_extent</i> <i>MPI_File_get_view</i> <i>MPI_File_iread_at</i> <i>MPI_File_iread</i> <i>MPI_File_iread_shared</i> <i>MPI_File_iwrite_at</i> <i>MPI_File_iwrite</i> <i>MPI_File_iwrite_shared</i> <i>MPI_File_open</i> <i>MPI_File_preallocate</i> <i>MPI_File_read_all_begin</i> <i>MPI_File_read_all_end</i> <i>MPI_File_read_all</i> <i>MPI_File_read_at_all_begin</i> <i>MPI_File_read_at_all_end</i> <i>MPI_File_read_at_all</i> <i>MPI_File_read_at</i> <i>MPI_File_read</i> <i>MPI_File_read_ordered_begin</i> <i>MPI_File_read_ordered_end</i> <i>MPI_File_read_ordered</i> <i>MPI_File_read_shared</i> <i>MPI_File_seek</i> <i>MPI_File_seek_shared</i> <i>MPI_File_set_atomicity</i> <i>MPI_File_set_info</i> <i>MPI_File_set_size</i> <i>MPI_File_set_view</i> <i>MPI_File_sync</i> <i>MPI_File_write_all_begin</i> <i>MPI_File_write_all_end</i> <i>MPI_File_write_all</i> <i>MPI_File_write_at_all_begin</i> <i>MPI_File_write_at_all_end</i>
<b>err</b> <i>MPI_Add_error_class</i> <i>MPI_Add_error_code</i> <i>MPI_Add_error_string</i> <i>MPI_Comm_call_errhandler</i> <i>MPI_Comm_create_errhandler</i> <i>MPI_Comm_get_errhandler</i> <i>MPI_Comm_set_errhandler</i> <i>MPI_Errhandler_free</i> <i>MPI_Error_class</i> <i>MPI_Error_string</i> <i>MPI_File_call_errhandler</i> <i>MPI_File_create_errhandler</i>	

<p> <i>MPI_File_write_at_all</i>  <i>MPI_File_write_at</i>  <i>MPI_File_write</i>  <i>MPI_File_write_ordered_begin</i>  <i>MPI_File_write_ordered_end</i>  <i>MPI_File_write_ordered</i>  <i>MPI_File_write_shared</i>  <i>MPI_Register_datarep</i> </p> <p><b>one2all</b></p> <p> <i>MPI_Bcast</i>  <i>MPI_Scatter</i>  <i>MPI_Scatterv</i> </p> <p><b>recv</b></p> <p> <i>MPI_Recv</i>  <i>MPI_Irecv</i>  <i>MPI_Recv_init</i>  <i>MPI_Probe</i>  <i>MPI_Iprobe</i> </p> <p><b>req</b></p> <p> <i>MPI_Start</i>  <i>MPI_Startall</i>  <i>MPI_Wait</i>  <i>MPI_Waitall</i>  <i>MPI_Waitany</i>  <i>MPI_Waitsome</i>  <i>MPI_Test</i>  <i>MPI_Testall</i>  <i>MPI_Testany</i>  <i>MPI_Testsome</i>  <i>MPI_Cancel</i>  <i>MPI_Grequest_start</i>  <i>MPI_Grequest_complete</i>  <i>MPI_Request_get_status</i>  <i>MPI_Request_free</i> </p> <p><b>rma</b></p> <p> <i>MPI_Accumulate</i>  <i>MPI_Get</i>  <i>MPI_Put</i>  <i>MPI_Win_complete</i>  <i>MPI_Win_create</i>  <i>MPI_Win_fence</i>  <i>MPI_Win_free</i>  <i>MPI_Win_get_group</i>  <i>MPI_Win_lock</i>  <i>MPI_Win_post</i>  <i>MPI_Win_start</i>  <i>MPI_Win_test</i>  <i>MPI_Win_unlock</i>  <i>MPI_Win_wait</i> </p> <p><b>scan</b></p> <p> <i>MPI_Exscan</i>  <i>MPI_Scan</i> </p> <p><b>send</b></p> <p> <i>MPI_Send</i>  <i>MPI_Bsend</i>  <i>MPI_Rsend</i>  <i>MPI_Ssend</i>  <i>MPI_Isend</i> </p>	<p> <i>MPI_Ibsend</i>  <i>MPI_Irsend</i>  <i>MPI_Issend</i>  <i>MPI_Send_init</i>  <i>MPI_Bsend_init</i>  <i>MPI_Rsend_init</i>  <i>MPI_Ssend_init</i> </p> <p><b>sendrecv</b></p> <p> <i>MPI_Sendrecv</i>  <i>MPI_Sendrecv_replace</i> </p> <p><b>serv</b></p> <p> <i>MPI_Alloc_mem</i>  <i>MPI_Free_mem</i>  <i>MPI_Buffer_attach</i>  <i>MPI_Buffer_detach</i>  <i>MPI_Op_create</i>  <i>MPI_Op_free</i> </p> <p><b>spawn</b></p> <p> <i>MPI_Close_port</i>  <i>MPI_Comm_accept</i>  <i>MPI_Comm_connect</i>  <i>MPI_Comm_disconnect</i>  <i>MPI_Comm_get_parent</i>  <i>MPI_Comm_join</i>  <i>MPI_Comm_spawn</i>  <i>MPI_Comm_spawn_multiple</i>  <i>MPI_Lookup_name</i>  <i>MPI_Open_port</i>  <i>MPI_Publish_name</i>  <i>MPI_Unpublish_name</i> </p> <p><b>status</b></p> <p> <i>MPI_Get_count</i>  <i>MPI_Status_set_elements</i>  <i>MPI_Status_set_cancelled</i>  <i>MPI_Test_cancelled</i> </p> <p><b>sync</b></p> <p> <i>MPI_Barrier</i> </p> <p><b>time</b></p> <p> <i>MPI_Wtick</i>  <i>MPI_Wtime</i> </p> <p><b>topo</b></p> <p> <i>MPI_Cart_coords</i>  <i>MPI_Cart_create</i>  <i>MPI_Cart_get</i>  <i>MPI_Cart_map</i>  <i>MPI_Cart_rank</i>  <i>MPI_Cart_shift</i>  <i>MPI_Cart_sub</i>  <i>MPI_Cartdim_get</i>  <i>MPI_Dims_create</i>  <i>MPI_Graph_create</i>  <i>MPI_Graph_get</i>  <i>MPI_Graph_map</i>  <i>MPI_Graph_neighbors</i>  <i>MPI_Graphdims_get</i>  <i>MPI_Graph_neighbors_count</i>  <i>MPI_Topo_test</i> </p>
---	--

<b>type</b> <i>MPI_Get_address</i> <i>MPI_Get_elements</i> <i>MPI_Pack</i> <i>MPI_Pack_external</i> <i>MPI_Pack_external_size</i> <i>MPI_Pack_size</i> <i>MPI_Type_commit</i> <i>MPI_Type_contiguous</i> <i>MPI_Type_create_darray</i> <i>MPI_Type_create_hindexed</i> <i>MPI_Type_create_hvector</i> <i>MPI_Type_create_indexed_block</i> <i>MPI_Type_create_resized</i> <i>MPI_Type_create_struct</i> <i>MPI_Type_create_subarray</i> <i>MPI_Type_dup</i> <i>MPI_Type_free</i> <i>MPI_Type_get_contents</i> <i>MPI_Type_get_envelope</i> <i>MPI_Type_get_extent</i> <i>MPI_Type_get_true_extent</i> <i>MPI_Type_indexed</i> <i>MPI_Type_size</i> <i>MPI_Type_vector</i> <i>MPI_Unpack_external</i> <i>MPI_Unpack</i>	
--	--

I\_MPI\_STATS\_ACCURACY

Use the I\_MPI\_STATS\_ACCURACY environment variable to decrease statistics output.

Syntax

I\_MPI\_STATS\_ACCURACY=<percentage>

Argument

<percentage>	Float threshold value
--------------	-----------------------

Description

Set this environment variable to collect data only on those MPI functions that take a larger portion of the elapsed time as a percentage of the total time spent inside all MPI calls.

Example

The following example represents a simple application code and IPM summary statistics format:

```
int main (int argc, char *argv[])
{
    int i, rank, size, nsend, nrecv;

    MPI_Init (&argc, &argv);

    MPI_Comm_rank (MPI_COMM_WORLD, &rank);
    nsend = rank;

    MPI_Wtime();

    for (i = 0; i < 200; i++)
    {
        MPI_Barrier(MPI_COMM_WORLD);
    }

    /* open "reduce" region for all processes */
    MPI_Pcontrol(1, "reduce");
    for (i = 0; i < 1000; i++)
        MPI_Reduce(&nsend, &nrecv, 1, MPI_INT, MPI_MAX, 0, MPI_COMM_WORLD);

    /* close "reduce" region */
```

```

MPI_Pcontrol(-1, "reduce");

if (rank == 0)
{
    /* "send" region for 0-th process only */
    MPI_Pcontrol(1, "send");
    MPI_Send(&nsend, 1, MPI_INT, 1, 1, MPI_COMM_WORLD);
    MPI_Pcontrol(-1, "send");
}
if (rank == 1)
{
    MPI_Recv(&nrecv, 1, MPI_INT, 0, 1, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
}

/* reopen "reduce" region */
MPI_Pcontrol(1, "reduce");
for (i = 0; i < 1000; i++)
    MPI_Reduce(&nsend, &nrecv, 1, MPI_INT, MPI_MAX, 0, MPI_COMM_WORLD);

MPI_Wtime();
MPI_Finalize ();

return 0;
}

```

Command:

```

mpiexec -n 4 -env I_MPI_STATS ipm:terse ./a.out

```

Stats output:

```

#####
#
# command : ./a.out (completed)
# host      : svlmpihead01/x86_64_Linux      mpi_tasks : 4 on 1 nodes
# start     : 05/25/11/05:44:13             wallclock : 0.092012 sec
# stop      : 05/25/11/05:44:13             %comm     : 98.94
# gbytes    : 0.00000e+00 total             gflop/sec  : NA
#
#####

```

Command:

```

mpiexec -n 4 -env I_MPI_STATS ipm ./a.out

```

Stats output:

```

#####
#
# command : ./a.out (completed)
# host      : svlmpihead01/x86_64_Linux      mpi_tasks : 4 on 1 nodes
# start     : 05/25/11/05:44:13             wallclock : 0.092012 sec
# stop      : 05/25/11/05:44:13             %comm     : 98.94
# gbytes    : 0.00000e+00 total             gflop/sec  : NA
#
#####
# region : *      [ntasks] = 4
#
#                               [total]      <avg>          min            max
# entries                      4              1              1              1
# wallclock                    0.332877        0.0832192      0.0732641      0.0920119
# user                         0.047992        0.011998      0.006999      0.019996
# system                       0.013997        0.00349925    0.002999      0.004
# mpi                         0.329348        0.082337      0.0723064      0.0912335
# %comm                       98.9398         98.6928      99.154
# gflop/sec                    NA              NA            NA            NA
# gbytes                       0              0            0            0
#
#                               [time]        [calls]        <%mpi>          <%wall>
# MPI_Init                     0.236192        4              71.71          70.95
# MPI_Reduce                    0.0608737      8000           18.48          18.29
# MPI_Barrier                  0.027415        800            8.32           8.24
# MPI_Recv                     0.00483489      1              1.47           1.45
# MPI_Send                     1.50204e-05     1              0.00           0.00
# MPI_Wtime                    1.21593e-05     8              0.00           0.00
# MPI_Finalize                 3.33786e-06     4              0.00           0.00

```

```

# MPI_Comm_rank      1.90735e-06      4      0.00      0.00
# MPI_TOTAL          0.329348      8822      100.00      98.94
#####
# region : reduce [ntasks] = 4
#
#      [total]      <avg>      min      max
# entries           8           2           2           2
# wallclock         0.0638561      0.015964      0.00714302      0.0238571
# user              0.034994      0.0087485      0.003999      0.015997
# system            0.003999      0.00099975      0           0.002999
# mpi               0.0608799      0.01522      0.00633883      0.0231845
# %comm             95.3392      88.7417      97.1808
# gflop/sec         NA           NA           NA           NA
# gbytes            0           0           0           0
#
#      [time]      [calls]      <%mpi>      <%wall>
# MPI_Reduce         0.0608737      8000      99.99      95.33
# MPI_Finalize       3.33786e-06      4           0.01      0.01
# MPI_Wtime          2.86102e-06      4           0.00      0.00
# MPI_TOTAL          0.0608799      8008      100.00      95.34
#####
# region : send [ntasks] = 4
#
#      [total]      <avg>      min      max
# entries           1           0           0           1
# wallclock         2.89876e-05      7.24691e-06      1e-06      2.59876e-05
# user              0           0           0           0
# system            0           0           0           0
# mpi               1.50204e-05      3.75509e-06      0           1.50204e-05
# %comm             51.8165      0           57.7982
# gflop/sec         NA           NA           NA           NA
# gbytes            0           0           0           0
#
#      [time]      [calls]      <%mpi>      <%wall>
# MPI_Send           1.50204e-05      1           100.00      51.82
#####
# region : ipm_noregion [ntasks] = 4
#
#      [total]      <avg>      min      max
# entries           13           3           3           4
# wallclock         0.26898      0.0672451      0.0661182      0.068152
# user              0.012998      0.0032495      0.001      0.004999
# system            0.009998      0.0024995      0           0.004
# mpi               0.268453      0.0671132      0.0659676      0.068049
# %comm             99.8039      99.7721      99.8489
# gflop/sec         NA           NA           NA           NA
# gbytes            0           0           0           0
#
#      [time]      [calls]      <%mpi>      <%wall>
# MPI_Init           0.236192      4           87.98      87.81
# MPI_Barrier        0.027415      800           10.21      10.19
# MPI_Recv           0.00483489      1           1.80      1.80
# MPI_Wtime          9.29832e-06      4           0.00      0.00
# MPI_Comm_rank      1.90735e-06      4           0.00      0.00
# MPI_TOTAL          0.268453      813           100.00      99.80
#####

```

# 5 Fault Tolerance

Intel® MPI Library provides extra functionality to enable fault tolerance support in the MPI applications. The MPI standard does not define behavior of MPI implementation if one or several processes of MPI application are abnormally aborted. By default, Intel® MPI Library aborts the whole application if any process stops.

Set the environment variable `I_MPI_FAULT_CONTINUE` to `on` to change this behavior. For example,

```
$ mpiexec -env I_MPI_FAULT_CONTINUE on -n 2 ./test
```

An application can continue working in the case of MPI processes an issue if the issue meets the following requirements:

- An application sets error handler `MPI_ERRORS_RETURN` to communicator `MPI_COMM_WORLD` (all new communicators inherit error handler from it)
- An application uses master-slave model and the application will be stopped only if the master is finished or does not respond
- An application uses only point-to-point communication between a master and a number of slaves. It does not use inter slave communication or MPI collective operations.
- Handle a certain MPI error code on a point-to-point operation with a particular failed slave rank for application to avoid further communication with this rank. The slave rank can be blocking/non-blocking send, receive, probe and test,
- Any communication operation can be used on subset communicator. If error appears in collective operation, any communication inside this communicator will be prohibited.
- Master failure means the job stops.

## 5.1 Environment Variables

### I\_MPI\_FAULT\_CONTINUE

Turn on/off support for fault tolerant applications.

#### Syntax

```
I_MPI_FAULT_CONTINUE=<arg>
```

#### Arguments

<arg>	Binary indicator
enable   yes   on   1	Turn on support for fault tolerant applications
disable   no   off   0	Turn off support for fault tolerant applications. This is default value

#### Description

Set this environment variable to provide support for fault tolerant applications.

## 5.2 Usage Model

An application sets `MPI_ERRORS_RETURN` error handler and checks return code after each communication call. If a communication call does not return, `MPI_SUCCESS` destination process should be marked unreachable and exclude communication with it. For example:

```
if(live_ranks[rank]) {  
    mpi_err = MPI_Send(buf, count, dtype, rank, tag, MPI_COMM_WORLD);  
    if(mpi_err != MPI_SUCCESS) {  
        live_ranks[rank] = 0;  
    }  
}
```

In the case of non-blocking communications, errors can appear during wait/test operations.

## 6 ILP64 Support

---

The term *ILP64* means that integer, long, and pointer data entities all occupy 8 bytes. This differs from the more conventional LP64 model in which only long and pointer data entities occupy 8 bytes while integer entities stay at 4 byte size. More information on the historical background and the programming model philosophy can be found for example in [http://www.unix.org/version2/whatsnew/lp64\\_wp.html](http://www.unix.org/version2/whatsnew/lp64_wp.html)

### 6.1 Using ILP64

Use the following options to enable the ILP64 interface

- Use the Fortran compiler driver option `-i8` for separate compilation and the `-ilp64` option for separate linkage. For example,
 

```
$ mpiifort -i8 -c test.f
$ mpiifort -ilp64 -o test test.o
```
- Use the `mpiexec -ilp64` option to preload the ILP64 interface. For example,
 

```
$ mpiexec -ilp64 -n 2 ./myprog
```

### 6.2 Known Issues and Limitations

- Data type counts and other arguments with values larger than  $2^{31}-1$  are not supported.
- Special MPI types `MPI_FLOAT_INT`, `MPI_DOUBLE_INT`, `MPI_LONG_INT`, `MPI_SHORT_INT`, `MPI_2INT`, `MPI_LONG_DOUBLE_INT`, `MPI_2INTEGER` are not changed and still use a 4-byte integer field.
- Predefined communicator attributes `MPI_APPNUM`, `MPI_HOST`, `MPI_IO`, `MPI_LASTUSED`, `MPI_TAG_UB`, `MPI_UNIVERSE_SIZE`, and `MPI_WTIME_IS_GLOBAL` are returned by the functions `MPI_GET_ATTR` and `MPI_COMM_GET_ATTR` as 4-byte integers. The same holds for the predefined attributes that may be attached to the window and file objects.
- Do not use the `-i8` option to compile MPI callback functions, such as error handling functions, user-defined reduction operations, etc.
- You have to use a special ITC library if you want to use the Intel® Trace Collector with the Intel MPI ILP64 executable files. If necessary, the Intel MPI `mpiifort` compiler driver will select the correct ITC library automatically.
- Use the `mpif.h` file instead of the MPI module in Fortran90\* applications. The Fortran module supports 32-bit `INTEGER` size only.
- There is currently no support for C and C++ applications.

## 7 Unified Memory Management

---

The Intel® MPI Library provides a way to replace the memory management subsystem by a user-defined package. You may optionally set the following function pointers:

- `i_malloc`
- `i_calloc`
- `i_realloc`
- `i_free`

These pointers also affect the C++ new and delete operators.

The respective standard C library functions are used by default.

The following contrived source code snippet illustrates the usage of the unified memory subsystem:

```
#include <i_malloc.h>
#include <my_malloc.h>

int main( int argc, int argv )
{
    // override normal pointers
    i_malloc = my_malloc;
    i_calloc = my_calloc;
    i_realloc = my_realloc;
    i_free = my_free;

#ifdef _WIN32
    // also override pointers used by DLLs
    i_malloc_dll = my_malloc;
    i_calloc_dll = my_calloc;
    i_realloc_dll = my_realloc;
    i_free_dll = my_free;
#endif

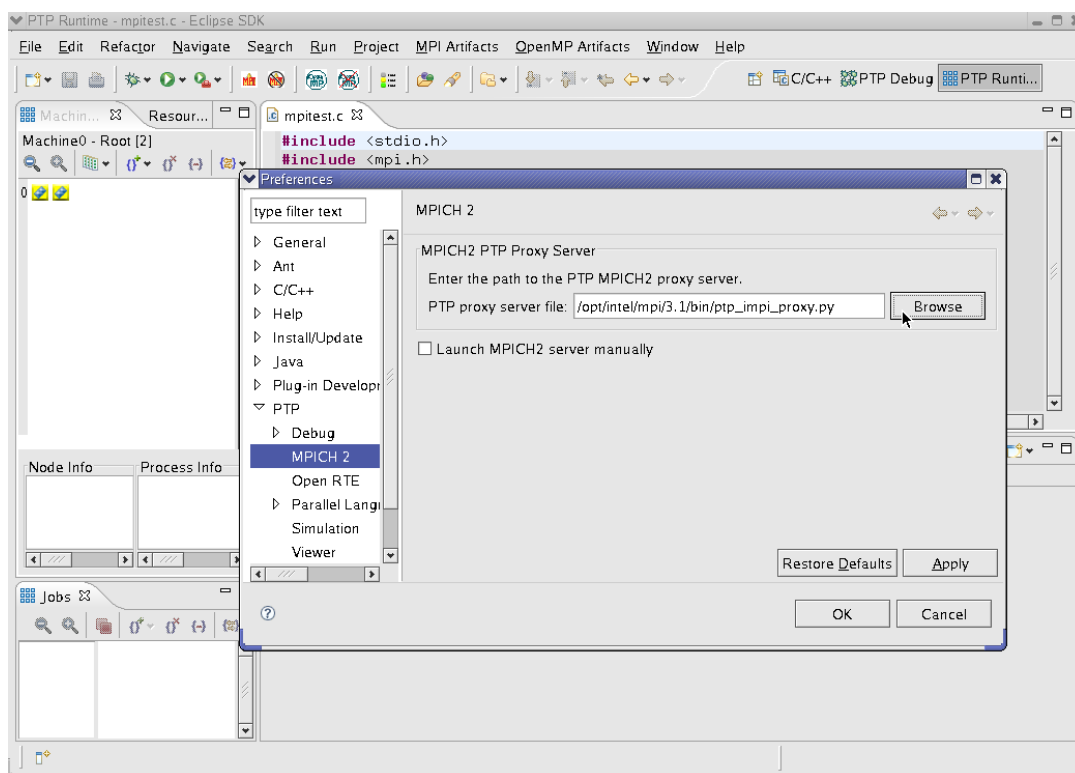
    // now start using Intel(R) libraries
}
```

## 8 Integration into Eclipse\* PTP

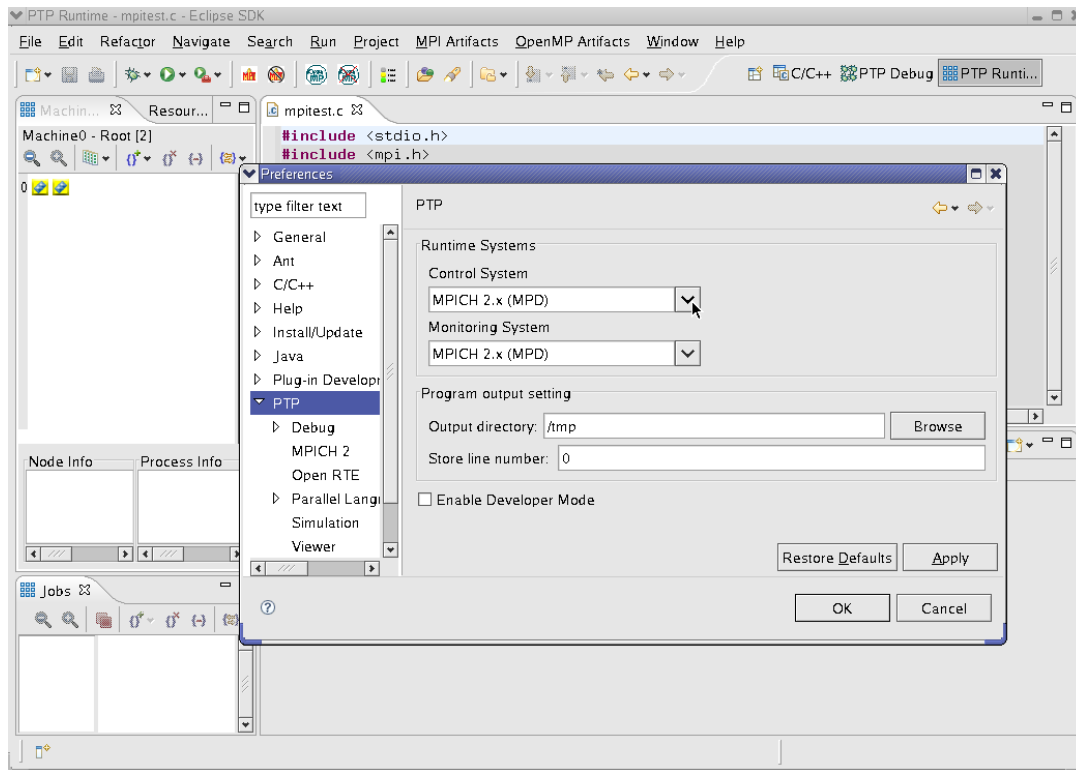
The Intel® MPI Library can be used with the Eclipse\* Parallel Tools Platform (PTP). You can launch parallel applications on the existing MPD ring from the Eclipse PTP graphical user interface. The MPD ring must be started prior to the PTP startup.

Perform the following configuration steps to use PTP with the Intel® MPI Library:

1. Set the `PTPPATH` environment variable to specify the location of the `ptplib.py` module.
2. Select Window->Preferences from the Eclipse main menu. Select PTP->MPICH 2 preference page.
3. Specify the full path to the `ptp_imp_i_proxy.py` file, for example, `<installdir>/bin/ptp_imp_i_proxy.py`. Click the **Apply** button.



4. Go to the PTP preference page.
5. Select MPICH2\* (MPD) in both Control System and Monitoring System drop down menus. If MPICH2\* (MPD) is already selected, click the **OK** button and restart Eclipse.



6. Switch to the PTP Runtime perspective.
7. In the Machines view you will see the cluster nodes on which the MPD ring is currently working.
8. Refer to the PTP User's Guide for more information. The PTP documentation is available at:  
<http://www.eclipse.org/ptp/doc.php>

## 9 Glossary

---

<b>hyper-threading technology</b>	A feature within the IA-32 family of processors, where each processor core provides the functionality of more than one logical processor.
<b>logical processor</b>	The basic modularity of processor hardware resource that allows a software executive (OS) to dispatch task or execute a thread context. Each logical processor can execute only one thread context at a time.
<b>multi-core processor</b>	A physical processor that contains more than one processor core.
<b>multi-processor platform</b>	A computer system made of two or more physical packages.
<b>processor core</b>	The circuitry that provides dedicated functionalities to decode, execute instructions, and transfer data between certain sub-systems in a physical package. A processor core may contain one or more logical processors.
<b>physical package</b>	The physical package of a microprocessor capable of executing one or more threads of software at the same time. Each physical package plugs into a physical socket. Each physical package may contain one or more processor cores.
<b>processor topology</b>	Hierarchical relationships of “shared vs. dedicated” hardware resources within a computing platform using physical package capable of one or more forms of hardware multi-threading.

# 10 Index

- 
- \$HOME/.mpd.conf, 60
  - [g]envexcl, 16
  - [g]envuser, 16
  - {cc,cxx,fc,f77,f90}=<compiler>, 12
  - l, 16
  - a, 16
  - a <alias>, 50
  - binding, 23
  - bootstrap <bootstrap server>, 23
  - bootstrap jmi, 23
  - bootstrap-exec <bootstrap server>, 23
  - branch-count <num>, 20
  - check\_mpi, 11, 14, 33, 55
  - check\_mpi [<checking\_library>], 19, 49
  - cleanup, 22
  - compchk, 12
  - config=<name>, 10
  - configfile <filename>, 20, 51
  - cpuinfo, 64
  - dapl, 27, 45
  - DAPL, 27, 45
  - demux, 35
  - demux <mode>, 21
  - disable-x, 21
  - dynamic\_log, 11
  - ecfn <filename>, 16, 51
  - echo, 11, 37, 38
  - Eclipse Parallel Tools Platform, 154
  - enable-x, 21
  - env <ENVVAR> <value>, 26, 51
  - envall, 26, 51
  - envexcl <list of env var names>, 51
  - envlist <list of env var names>, 19, 26, 51
  - envnone, 26, 51
  - envuser, 51
  - f <hostsfile>, 18
  - fast, 11
  - g, 11
  - g<l-option>, 48
  - gcc-version=<nnn>, 12
  - gdb, 50
  - gdba <jobid>, 50
  - genv, 48
  - genv <ENVVAR> <value>, 19, 48
  - genvall, 19, 48
  - genvexcl, 48
  - genvlist, 48
  - genvnone, 19, 48
  - genvnone, 48
  - genvuser, 48
  - gm, 28, 46
  - GM, 28, 46
  - grt <# of processes>, 19, 47
  - h, 38, 40, 41, 42, 46, 68
  - help, 37, 38, 39, 40, 41, 42, 43, 46, 68
  - help, 46
  - host <nodename>, 26, 52
  - hostfile <hostsfile>, 18
  - hosts <nodelist>, 21
  - Hydra, 16, 17, 18, 23, 34, 35
  - I\_MPI\_HYDRA\_JMI\_LIBRARY, 34
  - I\_MPI\_{CC,CXX,FC,F77,F90}, 14
  - I\_MPI\_{CC,CXX,FC,F77,F90}\_PROFILE, 13
  - I\_MPI\_ADJUST, 127
  - I\_MPI\_ADJUST\_<opname>, 127
  - I\_MPI\_ALLGATHER\_MSG, 132
  - I\_MPI\_ALLREDUCE\_MSG, 132
  - I\_MPI\_ALLTOALL\_MSG, 132
  - I\_MPI\_ALLTOALL\_NUM\_PROCS, 131
  - I\_MPI\_BCAST\_MSG, 131
  - I\_MPI\_BCAST\_NUM\_PROCS, 131
  - I\_MPI\_CACHE\_BYPASS, 94
  - I\_MPI\_CACHE\_BYPASS\_THRESHOLDS, 94
  - I\_MPI\_CHECK\_COMPILER, 14
  - I\_MPI\_CHECK\_PROFILE, 11, 14
  - I\_MPI\_COMPATIBILITY, 135
  - I\_MPI\_COMPILER\_CONFIG\_DIR, 15
  - I\_MPI\_CONN\_EVD\_QLEN, 107
  - I\_MPI\_DAPL\_BUFFER\_NUM, 104
  - I\_MPI\_DAPL\_BUFFER\_SIZE, 105, 118
  - I\_MPI\_DAPL\_CHECK\_MAX\_RDMA\_SIZE, 106, 107
  - I\_MPI\_DAPL\_CONN\_EVD\_SIZE, 107
  - I\_MPI\_DAPL\_DESIRED\_STATIC\_CONNECTIONS\_NUM, 109
  - I\_MPI\_DAPL\_DIRECT\_COPY\_THRESHOLD, 102
  - I\_MPI\_DAPL\_EAGER\_MESSAGE\_AGGREGATION, 103
  - I\_MPI\_DAPL\_MAX\_MSG\_SIZE, 107
  - I\_MPI\_DAPL\_PROVIDER, 88, 101
  - I\_MPI\_DAPL\_RDMA\_RNDV\_WRITE, 105
  - I\_MPI\_DAPL\_RNDV\_BUFFER\_ALIGNMENT, 105
  - I\_MPI\_DAPL\_SCALABLE\_PROGRESS, 104
  - I\_MPI\_DAPL\_SR\_BUF\_NUM, 108
  - I\_MPI\_DAPL\_SR\_THRESHOLD, 107

I\_MPI\_DAPL\_TRANSLATION\_CACHE, 101  
 I\_MPI\_DAPL\_TRANSLATION\_CACHE\_AVL\_TREE, 102  
 I\_MPI\_DAPL\_UD, 109, 112  
 I\_MPI\_DAPL\_UD\_ACK\_RECV\_POOL\_SIZE, 111, 116  
 I\_MPI\_DAPL\_UD\_ACK\_SEND\_POOL\_SIZE, 111, 116  
 I\_MPI\_DAPL\_UD\_CONN\_EVD\_SIZE, 112, 116  
 I\_MPI\_DAPL\_UD\_CONNECTION\_TIMEOUT, 116  
 I\_MPI\_DAPL\_UD\_CREATE\_CONN\_QUAL, 116  
 I\_MPI\_DAPL\_UD\_DESIRED\_STATIC\_CONNECTIONS\_NUM, 115, 116  
 I\_MPI\_DAPL\_UD\_DFACTOR, 116  
 I\_MPI\_DAPL\_UD\_DIRECT\_COPY\_THRESHOLD, 110, 114, 115, 116  
 I\_MPI\_DAPL\_UD\_EAGER\_DYNAMIC\_CONNECTION, 115, 116  
 I\_MPI\_DAPL\_UD\_FINALIZE\_RETRY\_COUNT, 116  
 I\_MPI\_DAPL\_UD\_FINALIZE\_TIMEOUT, 116  
 I\_MPI\_DAPL\_UD\_MAX\_MSG\_SIZE, 116  
 I\_MPI\_DAPL\_UD\_MAX\_RDMA\_DTOS, 117  
 I\_MPI\_DAPL\_UD\_MAX\_RDMA\_SIZE, 117  
 I\_MPI\_DAPL\_UD\_MULTIPLE\_EAGER\_SEND, 116  
 I\_MPI\_DAPL\_UD\_NA\_SBUF\_LIMIT, 116  
 I\_MPI\_DAPL\_UD\_NUMBER\_CREDIT\_UPDATE, 116  
 I\_MPI\_DAPL\_UD\_PKT\_LOSS\_OPTIMIZATION, 116  
 I\_MPI\_DAPL\_UD\_PORT, 116  
 I\_MPI\_DAPL\_UD\_PROVIDER, 88, 109, 116  
 I\_MPI\_DAPL\_UD\_RDMA\_MIXED, 115  
 I\_MPI\_DAPL\_UD\_RECV\_BUFFER\_NUM, 110, 116  
 I\_MPI\_DAPL\_UD\_RECV\_EVD\_SIZE, 113, 116  
 I\_MPI\_DAPL\_UD\_REQ\_EVD\_SIZE, 112, 116  
 I\_MPI\_DAPL\_UD\_REQUEST\_QUEUE\_SIZE, 116  
 I\_MPI\_DAPL\_UD\_RESENT\_TIMEOUT, 116  
 I\_MPI\_DAPL\_UD\_RNDV\_BUFFER\_ALIGNMENT, 113, 116  
 I\_MPI\_DAPL\_UD\_RNDV\_COPY\_ALIGNMENT\_THRESHOLD, 114, 116  
 I\_MPI\_DAPL\_UD\_RNDV\_DYNAMIC\_CONNECTION, 114, 116  
 I\_MPI\_DAPL\_UD\_RNDV\_MAX\_BLOCK\_LEN, 113  
 I\_MPI\_DAPL\_UD\_SEND\_BUFFER\_NUM, 110, 116  
 I\_MPI\_DAPL\_UD\_SEND\_BUFFER\_SIZE, 116  
 I\_MPI\_DAPL\_UD\_TRANSLATION\_CACHE, 111, 116  
 I\_MPI\_DAPL\_UD\_TRANSLATION\_CACHE\_AVL\_TREE, 112, 116  
 I\_MPI\_DAPL\_UD\_TRANSLATION\_CACHE\_MAX\_ENTRY\_NUM, 116  
 I\_MPI\_DAPL\_UD\_TRANSLATION\_CACHE\_MAX\_MEMORY\_SIZE, 116  
 I\_MPI\_DAT\_LIBRARY, 101  
 I\_MPI\_DEBUG, 11, 52  
 I\_MPI\_DEBUG\_OUTPUT, 53, 54  
 I\_MPI\_DEVICE, 52, 88, 89, 100  
 I\_MPI\_DYNAMIC\_CONNECTION, 93, 99  
 I\_MPI\_DYNAMIC\_CONNECTION\_MODE, 103  
 I\_MPI\_EAGER\_THRESHOLD, 91  
 I\_MPI\_EXTRA\_FILESYSTEM, 134  
 I\_MPI\_EXTRA\_FILESYSTEM\_LIST, 134, 135  
 I\_MPI\_FABRICS, 52, 88, 89, 90, 100  
 I\_MPI\_FABRICS\_LIST, 89, 90, 126  
 I\_MPI\_FALLBACK, 89, 90  
 I\_MPI\_FALLBACK\_DEVICE, 90  
 I\_MPI\_FAST\_COLLECTIVES, 130, 131  
 I\_MPI\_FAULT\_CONTINUE, 150  
 I\_MPI\_GATHER\_MSG, 134  
 I\_MPI\_HYDRA\_BOOTSTRAP, 23, 31  
 I\_MPI\_HYDRA\_BOOTSTRAP\_EXEC, 31  
 I\_MPI\_HYDRA\_BRANCH\_COUNT, 33  
 I\_MPI\_HYDRA\_CLEANUP, 35  
 I\_MPI\_HYDRA\_DEBUG, 26, 29  
 I\_MPI\_HYDRA\_DEMUX, 35  
 I\_MPI\_HYDRA\_ENV, 26, 29  
 I\_MPI\_HYDRA\_GDB\_REMOTE\_SHELL, 34  
 I\_MPI\_HYDRA\_HOST\_FILE, 28  
 I\_MPI\_HYDRA\_IDB\_TERMINAL, 20, 34  
 I\_MPI\_HYDRA\_IFACE, 35  
 I\_MPI\_HYDRA\_JMI\_LIBRARY, 23  
 I\_MPI\_HYDRA\_PMI\_AGGREGATE, 20, 34  
 I\_MPI\_HYDRA\_PMI\_CONNECT, 32  
 I\_MPI\_HYDRA\_RMK, 25, 31  
 I\_MPI\_INTRANODE\_DIRECT\_COPY, 91, 97  
 I\_MPI\_INTRANODE\_EAGER\_THRESHOLD, 91, 92, 97, 98, 100  
 I\_MPI\_INTRANODE\_SHMEM\_BYPASS, 100  
 I\_MPI\_JOB\_ABORT\_SIGNAL, 57  
 I\_MPI\_JOB\_CHECK\_LIBS, 33, 49, 55  
 I\_MPI\_JOB\_CONFIG\_FILE, 60  
 I\_MPI\_JOB\_CONTEXT, 41, 61  
 I\_MPI\_JOB\_FAST\_STARTUP, 59  
 I\_MPI\_JOB\_SIGNAL\_PROPAGATION, 30, 57  
 I\_MPI\_JOB\_STARTUP\_TIMEOUT, 56  
 I\_MPI\_JOB\_TAGGED\_PORT\_OUTPUT, 61  
 I\_MPI\_JOB\_TIMEOUT, 29, 30, 56  
 I\_MPI\_JOB\_TIMEOUT\_SIGNAL, 30, 57  
 I\_MPI\_JOB\_TRACE\_LIBS, 33, 48, 55  
 I\_MPI\_MPD\_CHECK\_PYTHON, 61, 62  
 I\_MPI\_MPD\_CLEAN\_LOG, 63  
 I\_MPI\_MPD\_RSH, 62  
 I\_MPI\_MPD\_TMPDIR, 62  
 I\_MPI\_MPIRUN\_CLEANUP, 17  
 I\_MPI\_MSG, 127  
 I\_MPI\_NETMASK, 117  
 I\_MPI\_OFA\_ADAPTER\_NAME, 121  
 I\_MPI\_OFA\_DYNAMIC\_QPS, 124  
 I\_MPI\_OFA\_LIBRARY, 124  
 I\_MPI\_OFA\_NONSWITCH\_CONF, 124

I\_MPI\_OFA\_NUM\_ADAPTERS, 120  
 I\_MPI\_OFA\_NUM\_PORTS, 121  
 I\_MPI\_OFA\_NUM\_RDMA\_CONNECTIONS, 121, 122  
 I\_MPI\_OFA\_PACKET\_SIZE, 124  
 I\_MPI\_OFA\_RAIL\_SCHEDULER, 122  
 I\_MPI\_OFA\_SWITCHING\_TO\_RDMA, 122  
 I\_MPI\_OFA\_TRANSLATION\_CACHE, 122  
 I\_MPI\_OFA\_TRANSLATION\_CACHE\_AVL\_TREE, 123  
 I\_MPI\_OFA\_USE\_XRC, 123  
 I\_MPI\_OUTPUT\_CHUNK\_SIZE, 58  
 I\_MPI\_PERHOST, 32, 54  
 I\_MPI\_PIN, 73, 76  
 I\_MPI\_PIN\_CELL, 78  
 I\_MPI\_PIN\_DOMAIN, 79, 80, 81, 86  
 I\_MPI\_PIN\_MODE, 73, 74  
 I\_MPI\_PIN\_ORDER, 86, 87  
 I\_MPI\_PIN\_PROCESSOR\_LIST, 74, 76, 79  
 I\_MPI\_PIN\_PROCS, 74  
 I\_MPI\_PMI\_EXTENSIONS, 58  
 I\_MPI\_PRINT\_VERSION, 55  
 I\_MPI\_PROCESS\_MANAGER, 17, 68  
 I\_MPI\_RDMA\_BUFFER\_NUM, 104  
 I\_MPI\_RDMA\_BUFFER\_SIZE, 105  
 I\_MPI\_RDMA\_CHECK\_MAX\_RDMA\_SIZE, 106  
 I\_MPI\_RDMA\_CONN\_EVD\_SIZE, 107  
 I\_MPI\_RDMA\_EAGER\_THRESHOLD, 102  
 I\_MPI\_RDMA\_MAX\_MSG\_SIZE, 107  
 I\_MPI\_RDMA\_RNDV\_BUF\_ALIGN, 105  
 I\_MPI\_RDMA\_RNDV\_BUFFER\_ALIGNMENT, 105  
 I\_MPI\_RDMA\_RNDV\_WRITE, 105  
 I\_MPI\_RDMA\_SCALABLE\_PROGRESS, 104  
 I\_MPI\_RDMA\_TRANSLATION\_CACHE, 101  
 I\_MPI\_RDMA\_VBUF\_TOTAL\_SIZE, 105  
 I\_MPI\_RDMA\_WRITE\_IMM, 108  
 I\_MPI\_REDS CAT\_MSG, 133  
 I\_MPI\_ROOT, 15  
 I\_MPI\_SCALABLE\_OPTIMIZATION, 92, 93  
 I\_MPI\_SCATTER\_MSG, 133  
 I\_MPI\_SHM\_BUFFER\_SIZE, 98  
 I\_MPI\_SHM\_BYPASS, 100  
 I\_MPI\_SHM\_CACHE\_BYPASS, 94  
 I\_MPI\_SHM\_CACHE\_BYPASS\_THRESHOLD, 94  
 I\_MPI\_SHM\_CACHE\_BYPASS\_THRESHOLDS, 94  
 I\_MPI\_SHM\_CELL\_NUM, 96  
 I\_MPI\_SHM\_CELL\_SIZE, 91, 96  
 I\_MPI\_SHM\_FBOX\_SIZE, 95, 96  
 I\_MPI\_SHM\_LMT, 97  
 I\_MPI\_SHM\_LMT\_BUFFER\_NUM, 97  
 I\_MPI\_SHM\_LMT\_BUFFER\_SIZE, 98  
 I\_MPI\_SHM\_NUM\_BUFFERS, 97  
 I\_MPI SOCK\_SCALABLE\_OPTIMIZATION, 92, 93  
 I\_MPI\_SPIN\_COUNT, 92, 100, 101  
 I\_MPI\_SSHM, 98  
 I\_MPI\_SSHM\_BUFFER\_NUM, 99  
 I\_MPI\_SSHM\_BUFFER\_SIZE, 99  
 I\_MPI\_STATS, 137, 142  
 I\_MPI\_STATS\_ACCURACY, 147  
 I\_MPI\_STATS\_BUCKETS, 138, 139  
 I\_MPI\_STATS\_FILE, 139, 143  
 I\_MPI\_STATS\_SCOPE, 137, 143  
 I\_MPI\_TCP\_BUFFER\_SIZE, 119  
 I\_MPI\_TCP\_NETMASK, 117  
 I\_MPI\_TCP\_POLLING\_MODE, 119  
 I\_MPI\_TIMER\_KIND, 136  
 I\_MPI\_TMI\_LIBRARY, 119  
 I\_MPI\_TMI\_PROBE\_INTERVAL, 120  
 I\_MPI\_TMI\_PROVIDER, 120  
 I\_MPI\_TMPDIR, 36  
 I\_MPI\_TRACE\_PROFILE, 10, 13, 14  
 I\_MPI\_USE\_DAPL\_INTRANODE, 100  
 I\_MPI\_USE\_DYNAMIC\_CONNECTIONS, 93  
 I\_MPI\_USE\_RENDEZVOUS\_RDMA\_WRITE, 105  
 I\_MPI\_WAIT\_MODE, 90, 93  
 -ib, 27, 45  
 -IB, 27, 45  
 -idb, 20, 49  
 IDB\_HOME, 20, 49, 59  
 -idba <jobid>, 49  
 -iface <interface>, 21  
 -ifhn <interface/hostname>, 16, 37, 50  
 -ilp64, 11  
 IPM, 142, 147  
 -l, 41, 42, 50  
 large message transfer (LMT), 97  
 LD\_LIBRARY\_PATH, 35  
 libjmi.so, 23  
 LMT, 97, 98  
 --locons, 16  
 -m, 16, 38, 39, 50  
 -machine <machine file>, 19  
 -machinefile <machine file>, 19, 47  
 max\_rdma\_size, 106  
 --maxbranch=<maxbranch> | -b <maxbranch>, 16  
 mpd, 17, 37, 38, 39, 40, 41, 42, 60, 61, 62, 74  
 MPD, 17  
 mpd.hosts, 39, 60  
 MPD\_SECRETWORD, 60  
 --mpd=<mpdcmd> | -m <mpdcmd>, 16  
 mpdallexit, 40, 63  
 mpdboot, 16, 17, 38, 39, 60, 61  
 mpdcheck, 41, 42  
 mpdcleanup, 40  
 mpdexit, 39

mpdhelp, 43  
 mpdkilljob, 43  
 mpdlistjobs, 42, 43  
 mpdringtest, 42  
 mpdsigjob, 43  
 mpdtrace, 17, 40, 41  
 MPI\_Allgather, 129, 132, 138  
 MPI\_Allgatherv, 129, 138  
 MPI\_Allreduce, 133, 138, 140  
 MPI\_Altoall, 129, 131, 132, 138  
 MPI\_Altoallv, 129, 138  
 MPI\_Altoallw, 129, 138  
 MPI\_ANY\_SOURCE, 119  
 MPI\_Barrier, 130, 138  
 MPI\_Bcast, 130, 131, 138  
 MPI\_COMM\_JOIN, 126  
 MPI\_COMM\_WORLD, 150  
 MPI\_ERRORS\_RETURN, 150, 151  
 MPI\_Exscan, 130, 138  
 MPI\_Finalize, 142  
 MPI\_Gather, 130, 134, 138  
 MPI\_Gatherv, 130, 138  
 MPI\_Init, 142  
 MPI\_Pcontrol, 142  
 MPI\_Reduce, 130, 138  
 MPI\_Reduce\_scatter, 130, 133  
 MPI\_Scan, 130, 138  
 MPI\_Scatter, 130, 133, 138  
 MPI\_Scatterv, 130, 138  
 mpicleanup, 35, 36, 37  
 mpiexec, 16, 17, 32, 44, 47, 49, 50, 51, 52, 54, 56, 57, 58, 71  
 mpiexec.hydra, 16, 17, 18, 20, 21, 22, 25, 29, 30, 32, 33, 36, 37  
 mpiicc -g, 54  
 mpirun, 16, 17, 35  
 mpitune, 69  
 mpitune, 22, 47, 67, 70  
 -mt\_mpi, 9  
 -mx, 28, 46  
 -MX, 28, 46  
 -n, 17, 19  
 -n <# of processes>, 26, 51  
 --ncpus=<ncpus>, 16  
 -noconf, 22, 50  
 -nolocal, 21, 47  
 -np, 17  
 -np <# of processes>, 26, 51  
 NUM\_RDMA\_BUFFER, 104  
 -O, 11  
 --ordered | -o, 16  
 -ordered-output, 22, 50  
 --parallel-startup | -p, 16  
 PATH, 9  
 -path <directory>, 22, 26, 52  
 -perhost, 18  
 -perhost <# of processes >, 19  
 -perhost <# of processes>, 47  
 PI\_Allreduce, 129  
 pmi\_proxy, 20  
 -pmi-aggregate, 20  
 -pmi-connect <mode>, 19  
 -pmi-noaggregate, 20  
 -ppn <# of processes >, 19  
 -ppn <# of processes>, 47  
 -print-all-exitcodes, 26  
 -print-rank-map, 26  
 -profile=<profile\_name>, 10, 13  
 -psm, 28, 46  
 -PSM, 28, 46  
 -rdma, 27, 45  
 -RDMA, 27, 45  
 RDMA\_IBA\_EAGER\_THRESHOLD, 102  
 --remcons, 16  
 -rmk <RMK>, 25  
 -rr, 18, 19, 47  
 -s <spec>, 22, 50  
 secretword, 60  
 -shell | -s, 16  
 -show, 11  
 -static, 10  
 -static\_mpi, 10  
 -t, 10  
 -t [<profiling\_library>], 19, 48  
 -tmi, 27, 45  
 -TMI, 27, 46  
 -tmpdir, 22  
 TMPDIR, 36, 62  
 TotalView, 20, 49  
 TOTALVIEW, 59  
 -trace, 10, 13, 33  
 -trace [<profiling\_library>], 19, 48  
 -tune, 21, 70, 71  
 -tune [<arg >], 46  
 -tv, 20, 49  
 -tva <jobid>, 49  
 -tva <pid>, 20  
 TVDSVRLAUNCHCMD, 49  
 -tvsu, 17, 49  
 -umask <umask>, 27, 52  
 --user=<user> | -u <user>, 16  
 -v, 12, 38, 41, 42  
 -V, 46  
 -verbose, 25, 38, 70

-version, 46

-version or -V, 22

VT\_ROOT, 10, 11, 15

-wdir <directory>, 26, 52